VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Demo scéna pro analýzu bio-signálů

Demo Scene for the Bio-signal Analysis

2015 Jakub Rodzenák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Jakub Rodzenák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Demo scéna pro analýzu bio-signálů
Demo Scene for the Bio-signal Analysis

Zásady pro vypracování:

Cílem práce je implementovat aplikaci, demo scénu, pro vizualizaci mozkové aktivity. Bude využito zařízení EMOTIV EPOC Headset. Zvolenou technologií je Microsoft .NET, OpenGL a jazyk C++.

1. Seznámení se s problematikou snímání a vizualizace EEG signálů. Přehled existujících technologií s ohledem na další použitelnost zařízení a software v nemedicínských aplikacích.
2. Návrh a implementace aplikace s požadovanou funkcionalitou.
3. Výkonnostní testy a experimenty s reálnými daty, uživateli.
4. Vyhodnocení experimentů a dosažených výsledků.

Seznam doporučené odborné literatury:

[1] John Kessenich: The OpenGL Shading Language, http://www.opengl.org/documentation/glsl/, 2011
[2] Mark Segal and Kurt Akeley: The OpenGL Graphics System: A Specification,
2011
[3] Saeid Sanei, J. A. Chambers: EEG Signal Processing, Wiley-Interscience; 1 edition (September 11, 2007), ISBN-13: 978-0470025819
[4] Edward Angel: Interactive Computer Graphics, ISBN-10:032153586 (2009)
[5] Jason Standers, Edward Kandrot: CUDA by Example: An Introduction to General-Purpose GPU Programming, ISBN-10: 0131387685, 2010

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Gajdoš, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.*

V Ostravě 19. dubna 2015 .....

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 19. dubna 2015 .....

Rád bych poděkoval Ing. Petrovi Gajdošovi, Ph.D. za vedení práce a Bc. Martinovi Bartečkovi za spolupráci na společné části této práce.

Abstrakt

Cílem této práce bylo vytvoření demo aplikace pro vizualizaci bio-signálů rozpoznávaných headsetem Emotiv EPOC. Součástí této práce bylo také seznámení se s problematikou BCI systémů a s headsetem samotným. Na této práci jsem spolupracoval s Bc. Martinem Bartečkem, který měl na starosti analýzu bio-signálů. Výsledkem jsou dvě aplikace, které obsluhují daný headset a představují možné použití daného BCI systému jako prostředek k rozpoznávání a následné vizualizaci pohybových akcí. Součástí byly výkonostní testy obou aplikací.

Klíčová slova: BCI, Emotiv, EPOC, EEG, vizualizace, OpenGL, Qt

Abstract

The aim of this work was to create a demo application for visualization of bio-signals Emotiv EPOC headset recognized. Part of this work was also to get familiar with the problems of BCI systems and headset itself. On this project I worked with Bc. Martin Barteček, who was in charge of the analysis of bio-signals. The results are two applications that control the headset and represent the possible use of the BCI system as a tool for recognition and visualization of the human motion events. Part of this thesis was also performance-testing of both applications.

Keywords: BCI, Emotiv, EPOC, EEG, visualization, OpenGL, Qt

Seznam použitých zkratek a symbolů

EEG	–	Electroencephalography
3D	–	Trojrozměrný
BCI	–	Brain - computer interface
Shader	–	program běžící na grafické kartě
Headset	–	náhlavní souprava
Fragment	–	Nejmenší část obrazu
GPU	–	Graphical processing unit
API	–	Application Programming Interface

Obsah

Seznam tabulek

Seznam obrázků

Seznam výpisů zdrojového kódu

1 Úvod

S objevem EEG signálů se otevřel zcela nový prostor pro bádání, a tak vznikla otázka, zda nelze tyto signály využít k rozpoznávání myšlenek, nálad a jiných mozkových aktivit. Tyto možnosti jsou dnes již nejen reálné, ale dostupnější než kdy dříve, díky relativně levným řešením od různých firem. Naskýtají se tak možnosti, které byly dříve výsadou pouze drahých výzkumných středisek a institucí. Přichází tak doba, která se přibližuje scénám, které byli doménou pouze vědeckofantastických knih a filmů. Vznikají tak tzv. BCI (Brain-Computer Interface), které propojují mozek a počítač pomocí headsetu, který snímá EEG signály a aplikace, která je zpracovává. Bohužel stále nelze číst lidské myšlenky, avšak je možné rozpoznávat různé činnosti spojené s pohybem svalů, náladami a psychickým stavem člověka popřípadě reagovat na intenzivní myšlenky. Lehce lze zjistit i míra aktivity v různých oblastech mozku. Tyto možnosti zkoumání EEG signálů nacházejí využití v různých odvětvích od lékařství až po zábavní průmysl. Analýza EEG signálů je již nedílnou součástí medicínských zařízení, avšak využití v zábavním průmyslu je stále spíše v experimentální fázi.

Cílem této práce bylo vytvoření aplikace pro vizualizaci pohybu lidského těla. Dále bylo náplní práce vytvořit aplikaci pro správu headsetu, nahrávání akcí a vizualizaci signálu EEG. Z těchto částí utvořit celek, který pokryje vše potřebné k rozpohybování modelu ve 3D scéně za pomocí rozpoznávání akcí v EEG signálu. Součástí práce bylo také měření výkonu aplikace při různých nastaveních vizualizace. Pro snímání byl použit headset EPOC od společnosti Emotiv. Ten poskytuje cenově dostupné řešení snímání EEG signálů.

V teoretické části práce je uveden úvod do teorie ohledně EEG a EEG signálu včetně typů vln nacházejících se v něm. Poté byl popsán mozek včetně jeho částí, hlavně důležitá část pro tuto práci, a to koncový mozek s jeho laloky. Dále jsou popsány BCI systémy z hlediska složení, historie a významných úspěchu experimentů jak na zvířatech, tak na lidech s pomocí různých typů snímacích zařízení. S tímto souvisí popis použitého headsetu Emotiv EPOC a standardizovaného systému rozložení elektrod.

V praktické části je jako první popsáno technologické zázemí implementace. Poté je popsána aplikace pro správu headsetu společně s popisem jednotlivých tříd a celkovou implementací programu. Dále je popsána druhá aplikace, a to demo scéna, včetně popisu implementace jednotlivých tříd pro načítání, práci s geometrií a kosterním systémem, vytváření animací lidské postavy a vizualizace scény s popisem vytvořených tříd a shaderů. Je zde také popsáno, jak tato aplikace zajišťuje výměnu informací mezi knihovnou pro zpracování dat z headsetu, touto aplikací a aplikací pro správu headsetu. Předposlední částí je, již dříve zmíněné, měření výkonu demo scény při různém nastavení. Poslední kapitolou je závěr shrnující nabyté poznatky. Celá implementace je v jazyce C++ s využitím knihovny OpenGL a frameworku Qt.

2 EEG

Je neurologická vyšetřovací metoda, díky níž můžeme sledovat aktivitu mozku v kýžených oblastech. Tato metoda je založena na faktu, že nervové buňky si předávají mezi sebou informace změnou elektrického napětí. Tyto změny lze na povrchu hlavy změřit elektrodami a získat tak kýženou mozkovou aktivitu ve sledované oblasti. Tato metoda je nazývána neinvazivní. Existují také invazivní a semi-invazivní metody. Ty sice podávají velmi přesné výsledky, avšak jsou pro běžné použití nevyhovující, jelikož je zapotřebí provést chirurgický zákrok pro získání příslušných hodnot elektrického napětí. Výhoda neinvazivní metody také spočívá v relativně nízkých nákladech na pořízení záznamu EEG signálu a fyzické nenáročnosti na zkoumanou osobu či zvíře [4].

Tato metoda byla vynalezena v roce 1875 Richardem Catonem[3], který poprvé změřil elektronické napětí mozkových hemisfér u králíků a opic. Později byla jeho práce dále rozvíjena dalšími vědci, až byl v roce 1924, německým neurologem Hansem Bergerem[2], sestrojen přístroj pro snímaní EEG signálu[1] a byl pořízen první záznam.

2.1 EEG signál

Je součet všech elektrických signálů mezi neurony v oblasti snímané elektrodou. Elektrická aktivita je dána přesunem elektricky nabitých iontů při změnách vodivosti buněčných membrán. A to konkrétně thalamu a kortexu, přičemž thalamus generuje rytmus těchto změn. Tento signál je závislý na čase, a tak jej lze jednoduše zobrazit v grafu jakožto hodnotu na ose Y. Těchto záznamů je obvykle pořizováno více na různých oblastech mozku. Záznamy jsou pořizovány sensory, které zachycují tzv. dipól, ten je tvořen dvěma stejně velkými opačně nabitými náboji[6][7].

Vyskytují se zde potenciály tohoto typu:

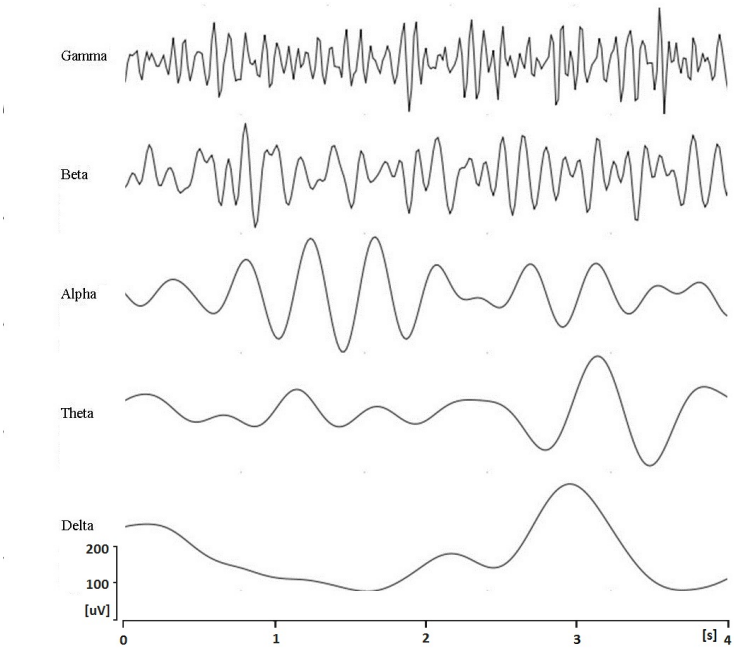
- **Jednotkové** Trvají v rámci jednotek milisekund. Snímány jsou na těle neuronů.
- **Synaptické** Trvají v rozmezí 15 - 45 ms
- **Dendritické** Jsou složitějšího tvaru a trvají až 100 ms .

Z těchto tří typů potenciálu se na EEG signálu nejvíce podílí synaptické a dendritické ve vrchních vrstvách kortexu. Mezi nejdůležitější vlastnosti signálu patří jeho amplituda, tvar, periodicita, rytmicita, perzistence, symetrie a reaktivita.

V EEG signálu se vyskytují vlny, které jsou definovány výše zmíněnými změnami potenciálu a tvoří tak mozkovou aktivitu (sled vln) v dané oblasti. U zdravého jedince se vlny pohybují do 80 Hz. Tyto vlny mohou být různých délek, a tak rozlišujeme pomalé a rychlé vlny. Mezi pomalé řadíme vlny s délkou menší než 8Hz a zbytek jsou klasifikovány jako rychlé vlny. Amplituda signálu se u zdravého jedince vyskytuje do 300 μV. Vlny se dále dělí do podskupin pojmenovaných podle řecké abecedy, seřazeny tak, jak byly v historii objeveny. Liší se amplitudou, významem i polohou, kde jsou tyto vlny rozpoznávány. V následujících odrážkách jsou popsány jejich typy[9][1].

Obrázek 1: Ukázka náhlavní soupravy pro snímání EEG signálu [27]

- **Alpha**: je nejvýznamnější aktivitou v EEG. Tyto vlny se pohybují ve frekvenčním pásmu od 8-14 Hz. Amplituda může být různá, avšak u dospělých a zdravých jedinců se pohybuje maximálně do 50 μV. Tyto vlny byly objeveny historicky nejdříve a byly jako první zaznamenány výše zmíněným Hansem Bergerem. Vyskytují se při uvolněném stavu, v leže, při zavřených očích v bdělosti. Po otevření očí se tyto vlny utlumují. Maximální amplituda vln je měřena v parietooccipitální oblasti, kdy rytmus je ovlivněný talamickými strukturami.
- **Beta**: je nejběžnější aktivitou a objevuje se při otevřených očích, při běžné aktivitě jako je rozhodování, přemýšlení a obecně při vstřebávání informací z okolního světa. Je nejvýraznější na frontálním laloku. Pohybuje se v poměrně širokém frekvenčním pásmu, a to od 14 Hz po maximálně 30 Hz. Výška amplitudy nedosahuje více než 20 μV. Dlouhodobá rytmická aktivita je spojena s užíváním léku či nemocemi. Dále se dělí na tři podskupiny podle frekvence.
  - **Nízká beta**: 12 - 15 Hz. Soustředění směrované dovnitř.
  - **Střední beta**: 18 - 18 Hz. Stav bdělosti, soustředění ven.
  - **Vysoká beta**: nad 18 Hz. Podrážděnost a ostražitost.
- **Gama**: má nejvyšší pásmo a to nad 30 Hz s amplitudou od 3 μV do 10 μV. Vlny tohoto typu se vyskytují při extrémním soustředění, řešením problémů, logickém uvažování, ale i vypětí a stresu. Tato aktivita je zaznamenávána hlavně v neokortexu, tektu a čichovém bulbu.
- **Delta**: jsou to vlny s nejnižší frekvencí, a to v rozsahu od 0,5 do 4 Hz. Odpovídají stavu velmi hlubokého spánku nebo např. hypnóze. Z vývojového hlediska je to aktivita snímána v kojeneckém věku. U bdělého, dospělého člověka je tento stav



Obrázek 2: Ukázka typů mozkových vln seřazených podle frekvence

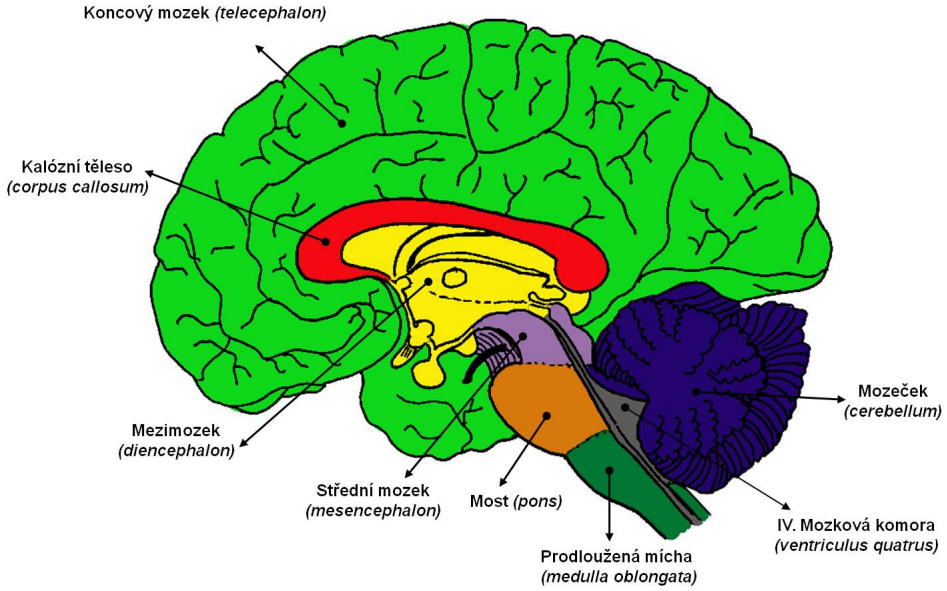
považován za patologický. Amplituda je největší ze všech a pohybuje se v rozmezí 20 - 200 μV.

- **Theta**: jsou typické vlny zrajícího mozku dětí v bdělém stavu, nejvýraznější v 2 - 3 letech. Jsou asociovány se sněním, představivostí a živými vzpomínkami. Zasahují do frekvenčního pásma 5 - 7 Hz s amplitudou 5 - 100 μV. Vlny jsou lokalizovány v parentální a temporální oblasti mozku.

U dětí a starých lidí se EEG signál v některých ohledech liší. U dětí se výrazně mění EEG signál od narození až do šesti let a až teprve v 15 - 20 letech se tento signál stabilizuje. Rozdílnosti se projevují hlavně ve frekvenci, aktivitě a změnou tvaru vln a reaktivností. Alpha aktivita bývá dominantní, beta aktivita se rozšiřuje a naopak výskyt theta vln se snižuje. U starých lidí se pak alpha aktivita zpomaluje, Beta se zmnožuje a zvětšuje se výskyt vln delta a theta. Jednotlivé typy vln ukazuje obrázek [2].

2.2 Mozek

Je jeden z nejdůležitějších orgánů lidského těla a zároveň hlavní testovací subjekt této práce. Mozek je řídícím a organizačním centrem nervové soustavy a je uložen v neurokraniu. Hmotnost mozku je přibližně okolo 1500 g u dospělého člověka, ovšem můžeme se setkat i s hodnotami výrazně se lišícími. Nejlehčí mozek tak měl 241 g a naopak nejtěžší



Obrázek 3: Mozek a jeho hlavní části[11]

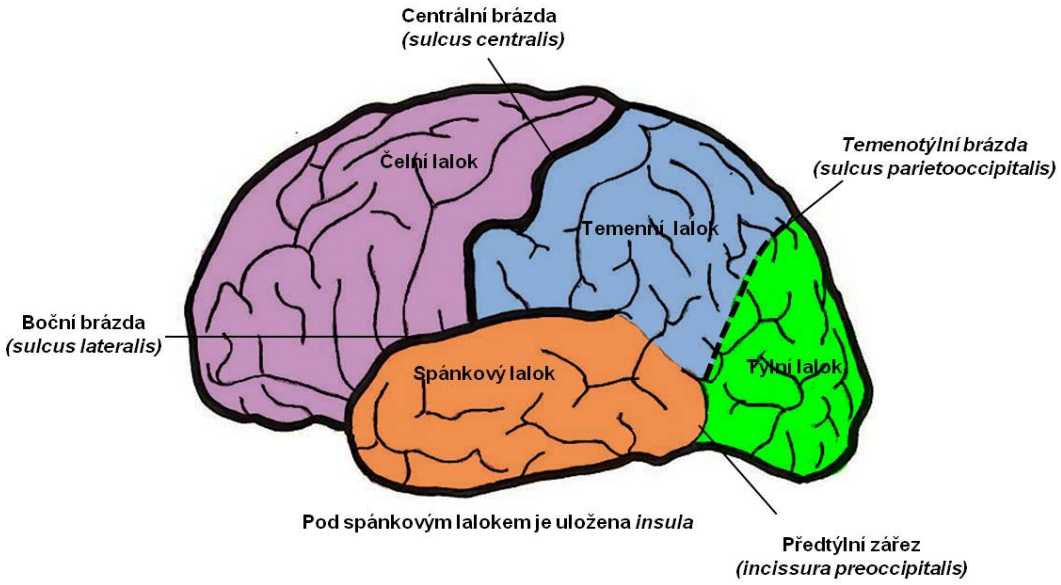
vážil okolo 2800 g. Oba tyto případy se však vyskytovali u jedinců s idiocií. Mozek obsahuje až 100 miliard neuronů, které komunikují přes více než biliardu spojení zvanými synapse.

Mozek je dělen na několik částí plnící různou funkcionalitu, ty spolupracují a tvoří tak hlavní orgán [3][11]. Mezi základní části patří zadní mozek, střední mozek a přední mozek. Zadní mozek je tvořen prodlouženou míchou, Valorovým mostem a mozečkem. Přední mozek se skládá z mezimozku a koncového mozku.

Prodloužená mícha, Valorův most a střední mozek tvoří nejstarší část mozku nazývanou mozkový kmen. Dále je mozek chráněn, před vnějšími vlivy lebkou, mozkovými blánami a mozkomíšním mokem. Nervové buňky jsou také chráněny, a to pomocí hematoencefalické bariéry zkráceně HEB. Ta je tvořena souvislou vrstvou endotelových buněk, která zamezuje průchodu látek mezibuněčným prostorem a chrání nervové buňky před škodlivinami z krve. V následující podkapitole bude popsána pouze jedna část mozku, a to koncový mozek, který je pro tuto práci nejdůležitější, jelikož je centrem lidského vědomí.

2.2.1 Koncový mozek

Je hlavní součástí centrálního nervového systému a představuje 83% mozkové hmoty. Na povrchu se nachází šedá kůra mozková, latinsky *cortex celebri*. Dále je tvořen bílou hmotou, která je pod kortexem, a bazálními ganglii, neboli shluky šedé hmoty uvnitř té bílé. Mozek je tvořen dvěma hemisférami - levou a pravou, mezi nimiž je sagitálně orientovaná podélná mozková štěrbina. Obě mozkové hemisféry spojuje mozkový svalek



Obrázek 4: Rozmístění mozkových laloků[11]

taktéž zvaný "telefonní ústředna hemisfér", který koordinuje obě hemisféry. Na povrchu mozku se nacházejí rýhy, zvané sulci, a mezi nimi jsou závity, zvané gyri. Na hemisférách se rovněž nacházejí brázdy, které mozek rozdělují do laloků.

2.2.2 Mozkové laloky

Celkem je pět laloků oddělených konstantními brázdami [4]. V přední části mozku se nachází čelní lalok, který sahá až po centrální brázdu. Za ním je temenní lalok, který leží za centrální brázdou a pokračuje až po temenotýlní brázdu. V zadní části je lokalizována týlní oblast. Částečně pod týlním a temenním lalokem je spánkový lalok, který je oddělen boční brázdou. Posledním lalokem je ostrovní lalok, který je skryt pod čelním, temenním a spánkovým lalokem pod boční a centrální brázdou.

- **Čelní lalok**
  - *Primární motorická oblast*: nachází se pod centrální brázdou. Stará se o zručné a přesné pohyby těla a to především prstů, předloktí a svalů na tváři.
  - *Premotorická kůra*: kontroluje složitější pohyby závislé na zpětné vazbě např. uchopování předmětů, skok přes překážky apod.
  - *Čelní zraková oblast*: má na starosti rychlé pohyby očí při sledování pohybujícího se objektu.
  - *Primární čichová oblast*: je lokalizována po stranách čelního laloku a je zodpovědná za poznávání pachů či vůní. Je napojena na čichové oblasti v nižších centrech limbického systému.

  - *Brocovo řečové centrum*: závisí na dominanci hemisfér. V levé dominantní hemisféře řídí tvorbu řeči, zatímco v pravé kontroluje emoční zabarvení a krátkodobé paměti slov a řeči. U dominantní pravé hemisféry se tyto funkce prohodí. Dále toto centrum souvisí s levorukostí a pravorukostí.
  - *Prefrontální kůra*: podílí se na myšlení, vnímání, zapamatování a rozpoznávání informací, sebeuvědomování, abstraktním myšlení a sebekontrole.

- **Temenní lalok**
  - *Senzitivní korová oblast*: stará se o vnímaní chutí, dotyku, bolesti, chladu a tepla. Také zajišťuje lokalizaci prostorových vjemů. Nachází se hned za centrální brázdou.
  - *Somatosenzitivní asociační oblast*: nachází se za senzitivní korovou oblastí a plní funkci asociace tvarů předmětu na základě předchozích zkušeností.

- **Týlní lalok**
  - *Primární zraková oblast*: nachází se na samém kraji týlního laloku. Přijímá informace ze sítnice očí, a poté tyto informace zpracovává pro obě oči dohromady.
  - *Asociační zraková oblast*: spolupracuje s primární zrakovou oblastí na rozpoznávání barev, tvaru a pohybu objektu. Spolupracuje s ostatními oblastmi mozku pro např. rozpoznávání textu z obrazu či rozpoznávání obličejů. Komunikuje s těmito oblastmi pomocí přední a zadní dráhy. Nachází se v přední části týlního laloku.

- **Spánkový lalok**
  - *Sluchová oblast a vestibulární oblast*: stejně jako u týlního laloku se zde rozlišují primární a asociační oblasti. Primární oblast rozpoznává hlasitost, rytmus a výšku tónu, zatímco asociační oblast těmto rozpoznaným zvukům přiřazuje zvuky z paměti např. hudba, hřmění, jásot.

- **Ostrovní lalok**
  Funkce tohoto laloku jsou nejméně prozkoumané. Souvisejí však hlavně s řízením vegetativních funkcí a emotivitou. Účastní se tak na tvorbě emocí, hlavně na pocitu líbí se mi/nelíbí se mi a komplexními funkcemi tzv. emoční inteligence.
  - *Přední část* řídí vegetativní reakce na základě čichových a chuťových počitků. Aktivita této části je velmi ovlivněna expozicí nelibým pachům.
  - *Zadní část* zpracovává informace především ze sluchových, premotorických, a somatosenzorických oblastí mozku.

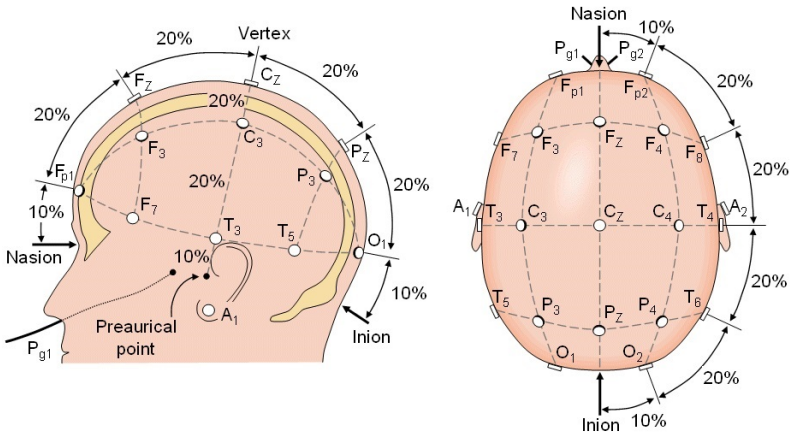
3 Snímání mozkové aktivity

Jak již bylo řečeno dříve, existují tři metody snímání EEG signálu. Invazivní, semi-invazivní a neinvazivní. První dvě metody mohou být pro zkoumaný subjekt velmi nebezpečné, protože hrozí krvácení nebo zjizvení mozkové tkáně s trvalými následky. Proto se nejčastěji využívá metoda neinvazivní, která nemá žádné škodlivé účinky a je méně náročná jak časově, tak i co se týče nákladů na zařízení. Dnes si lze pořídit již několik zřízení využívající tuto metodu a lišící se moha parametry např. počtem a kvalitou elektrod, jejich umístěním a kvalitou výstupu zesilovače. Právě použití kvalitních materiálů a citlivých zesilovačů nejvíce zvyšuje přesnost výsledného signálu. Snímaný signál také ovlivňují elektromagnetické vlny rádiových a televizních vysílání nebo mobilních sítí. Ideální podmínky(odstínění místnosti) je však v mnoha situacích obtížné realizovat, a tak se většinou zanedbávají. Všechny neinvazivní zařízení však zpravidla obsahují následující části.

- **Zesilovač**: EEG signál se pohybuje v desítkách, maximálně stovkách μV, proto se musí před samotnou analýzou zesílit. Jsou použity dva diferenciální zesilovače. Prvně jde signál přes předzesilovač, a poté na vstup hlavního zesilovače. Každý jej zesílí přibližně tisíckrát. Předzesilovač by se měl nacházet co nejblíže k elektrodám, aby docházelo k co nejmenšímu zkreslení signálu z elektrod.
- **Filtry**: Vstupní signál je třeba ořezat o nežádoucí frekvence. Před a za výše zmíněný zesilovač se aplikují dva filtry. Před ním najde uplatnění horní propust pro signály menší než 0,5 Hz a stejnosměrnou složku. Za zesilovač se naopak používá dolní propust, která naopak omezí maximální frekvenci většinou na 70 Hz. Kromě těchto frekvencí se pomocí tzv. notch filtru filtruje také frekvence 50 Hz, jakožto rušení z elektrické sítě.
- **Elektrody**: Slouží k snímání napěťových potenciálů na povrchu lebky. Jsou vyrobeny z kovu, a jeho druh a kvalita ovlivňuje výraznou měrou kvalitu snímaného signálu. Používá se většinou čisté stříbro, nerezová ocel, pozlacené stříbro anebo čisté zlato. Elektrody jsou rozmístěny na různých místech lebky tak, aby snímaly aktivitu v kýžených oblastech mozku. Toto rozmístění bylo sjednoceno systémem 10/20 v roce 1957 na mezinárodním kongresu EEG. Tento systém je popsán v další kapitole.

3.1 Systém 10/20

Tento systém slouží k mapování pozic sensorů na lebku[25]. Je standardizovaným systémem rozmístění elektrod a byl zaveden kvůli porovnatelnosti naměřených výsledků napříč studiemi subjektů. Mapování elektrod je založeno na vztahu pozic těchto elektrod k mozkové kůře. Číslice 10 a 20 v názvu odpovídají rozmezí elektrod v celkovém pokrytí lebky jak v příčném, tak podélném směru viz obrázek[5]. Takto je definováno základních 19 pozic. Pro vědecké účely je však tento počet nedostačující, a tak byl tento systém podle stejné konvence rozšířen na 10/10 a 10/5. Maximální počet elektrod tak vzrostl na 128. Tento počet limituje také velikost sensorů a lebky samotné. Písmena v názvu jednotlivých



Obrázek 5: Rozložení elektrod na lebce podle systému 10/20 s vyznačeným procentuálním rozmístěním. Vlevo pohled z profilu, vpravo pohled z vrchu[25]

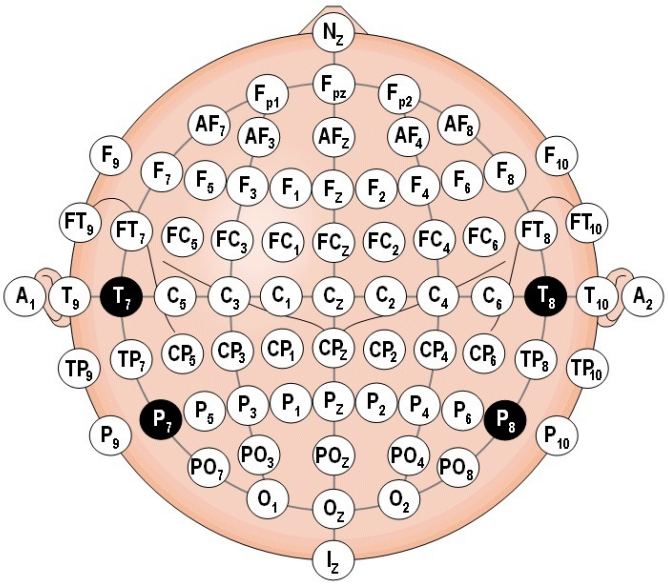
sensorů jsou odvozena z oblasti lebky kde se nacházejí. Číselné označení je rozlišeno na sudé a liché reprezentující levou a pravou hemisféru. Sensory umístěné na střední linii jsou označovány místo číslicí písmenem z. Níže jsou rozvedeny názvy oblastí podle jednotlivých písmen.

- **C** – centrální oblast
- **P** – parientální (temenní lalok)
- **F** – frontální (čelní lalok)
- **Fp** – frontopolární (čelní lalok)
- **O** – okcipitální (týlní lalok)
- **T** – temporální oblast (spánková)

Alternativu k tomuto systému nabízí společnosti specializující se na výrobu náhlavních souprav určených k diagnostice různých nemocí či výzkumů zaměřených na určité oblasti mozku. Lze se tak setkat s náhlavními soupravami s vyšším počtem elektrod v oblasti řečových center a podobně. Mimo tyto účelově cílené systémy jsou použity i systémy sférického rozmístění založené na stejné vzdálenosti elektrod od sebe. Tyto přístroje vyrábí např. firma *EasyCap*[24].

3.2 Emotiv EPOC

Je náhlavní souprava(headset)[23] snímající EEG aktivitu mozku od společnosti Emotiv[7]. Emotiv je společnost původem z Austrálie zabývající se vývojem počítačového rozhraní založeného na EEG technologii. Firma Emotiv[22] byla založena v roce 2003



Obrázek 6: Vrchní pohled na rozmístění a pojmenování elektrod podle systému 10/10[25]

těmito vědci: neurologem a profesorem Allanem Snyderem, Neilem Westem a technologickými podnikateli Tan Le a Nam Do.

Toto zařízení využívá neinvazivní způsob snímání mozkové aktivity. Disponuje bezdrátovou komunikací(2.4 GHz), a tak jej lze snadno propojit s počítačem za pomocí USB přijímače. Obsahuje také vlastní baterii, které výrobce deklaruje až 12 hodin provozu na jedno nabití. Nabíjení probíhá za pomocí USB kabelu propojeného s počítačem. Bohužel není možné zároveň nabíjet zařízení a používat jej. EPOC obsahuje rovněž gyroskopický sensor pro snímání pohybu hlavy ve dvou osách. Headset je vybaven 14-ti sensory pro snímání aktivity v jednotlivých oblastech podle systému 10/20 popsaném v minulé kapitole. Obsahuje také dva referenční sensory. Kvalitu přijmu signálu jednotlivých sensorů posílá headset společně s příslušnými daty. Sensory jsou opatřeny tkaninou, kterou je potřeba před použitím navlhčit fyziologickým roztokem dodaným rovněž ve standardním balení headsetu. Všechny sensory jsou vyměnitelné a součástí balení je i pouzdro pro jejich uskladnění. K headsetu je dodávána také sada aplikací pro správu headsetu a demonstrování jeho možností. Jedná se o TestBench, EmoComposer, EmoKey, hra Spirit Mountain a další. Dále je k dispozici SDK pro možnosti implementace vlastních aplikací využívající Emotiv EPOC. SDK je tvořeno knihovnami *edk.dll* a *edk_utils.dll*, které obsahují vše potřebné pro obsluhu a získání dat z headsetu. Knihovna *edk.dll* obsahuje třídu EmoEngine, která se stará o schraňovaní dat a informací z headsetu, a tyto data zpracovává do srozumitelné formy. Společnost Emotiv nenabízí aktualizace svého SDK zdarma, což znamená, že každou novou verzi je nutné zakoupit znovu. SDK je dostupné pro systémy Windows, Linux a Apple Mac OS X. V následující tabulce je specifikace headsetu Emotiv EPOC dle oficiálních stránek výrobce[3.2].

Obrázek 7: Headset Emotiv EPOC[23]

Parametery	
Počet kanálů	14 + 2 (referenční CMS/DRL, lokalizační P3/P4)
Názvy kanálů (10/20)	AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8,AF4
Vzorkovací metoda	Sekvenční vzorkování, analogově digitální převodník
Vzorkovací frekvence	128 vzorků za vteřinu, 2048 Hz vnitřně
Rozlišení	14 bitů, 1 LSB = 0,51 uV
Šířka pásma	0,2–45 Hz, digitální notch filtr na 50 Hz a 60 Hz
Filtry	zabudovaný digitální 5-ti řadý sinc filtr (dolní propust)
Dynamický rozsah	8400 uV (pp)
Spojovací mód	AC spojený
Spojení	proprietární bezdrátové, pásmo 2,4 GHz
Napájení	LiPoly
Výdrž na baterii	12 hodin
Měření impedance	kontrola kvality spojení v reálném čase pomocí patentovaného systému

Tabulka 1: Parametry Emotiv EPOC

4 BCI

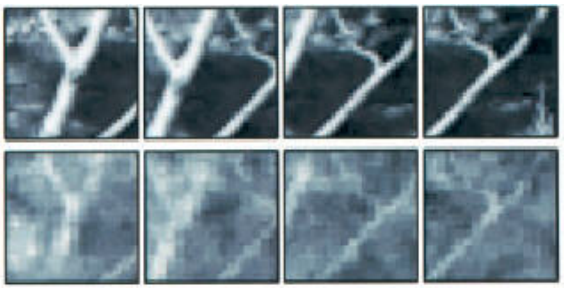
Jako BCI označujeme, jak již z překladu zkratky vyplývá, propojení mezi mozkem a počítačem[10]. Počátek vývoje těchto systémů se datuje do 70. let minulého století do města Los Angeles. První pokusy sestrojit toto rozhraní se zaměřují na čistě medicínské účely, a to především na alespoň částečné obnovení sluchových, zrakových či pohybových schopností subjektu pomocí implantátů. První testy byly prováděny na zvířatech. V 90 letech začal výzkum na lidech. Díky velké přizpůsobivosti mozku byly subjekty schopny, po adaptaci, využívat implantátů přirozeně a bez větších problémů.

4.1 Výzkum prováděný na zvířatech

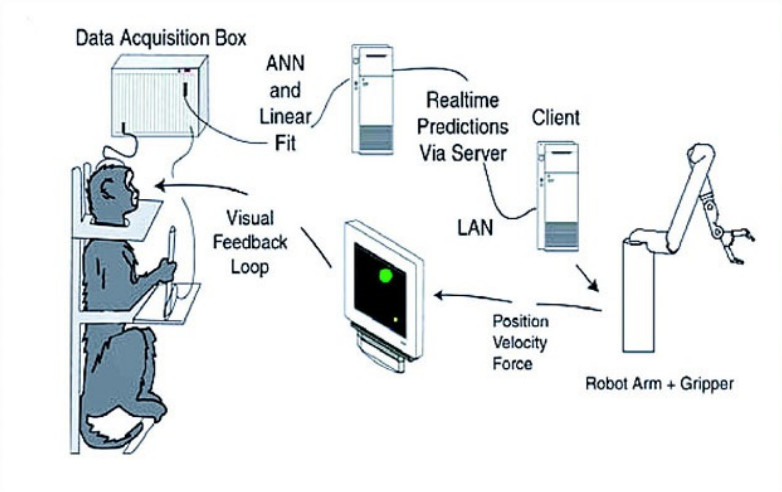
Několik vědeckých laboratoří bylo schopno snímat mozkovou aktivitu opic a krys invazivní metodou k ovládání BCI systému. Opice tak byly schopné ovládat kurzor myši na počítači nebo např. pohybovat robotickou rukou a uchopovat různé předměty. V roce 1969 přišli vědci na skutečnost, že jsou opice schopny se naučit ovládat robotickou ruku na základě svých myšlenek. Později bylo dokázáno, že tyto opice jsou schopné generovat požadované vzorce v mozkové aktivitě, pokud jsou za to posléze odměňovány. V 70. letech byl také vynalezen algoritmus pro rekonstruování pohybu na základě myšlenek. O dekádu později Apostolos Georgopuolos objevil matematické závislosti mezi elektrickými impulzy kortikální motorické oblasti mozku opic druhu makak rhesus a pohybem jejich rukou na základě kosinové funkce.

Přední úspěchy vědeckých výzkumů:

- **Phillip Kennedy**: Se svými kolegy sestrojili první BCI implantováním elektrod do mozku opic.
- **Yang Dan**: Jako první dokázal zpracovat výboje neuronů produkovaných při vytváření obrazu ve zraku koček. On a jeho tým použili pole elektrod implantovaných do thalamu ostře vidící kočky. Kočce poté bylo ukázáno osm krátkých videí a mezí tím byly nahrávány tyto výboje v thalamu. Poté byly tyto záznamy, pomocí matematických filtrů, dekódovány a zrekonstruovány do obrazu, ve kterém pak bylo možné rozpoznat jak určité scény, tak i pohybující se objekty[8].
- **Miguel Nicolelis**: Používal, na rozdíl od ostatních, několika elektrod k snímání aktivity ve velkých částech mozku pro použití v systémech BCI. Zredukoval tak variabilitu výstupního signálu při použití jedné elektrody. První pokusy prováděl na opicích z čeledi ploskonosých, u kterých v roce 2000 dokázal úspěšně reprodukovat pohyby rukou, když používaly opice joystick nebo se krmily. BCI fungovalo v reálném čase a zároveň byla robotická ruka ovládána vzdáleně pomocí internetového protokolu [9]. Tento způsob se nazývá otevřená smyčka. V dalším výzkumu Miguel použil jiný druh opic, a to druh makak rhesus, který měl lepší předpoklady pro snímáni pohybové aktivity. Zaměřil se, a byl úspěšný, na postup zvaný uzavřená smyčka, kdy byly opice schopny ovládat robotickou ruku umístěnou vedle



Obrázek 8: Rekonstruovaný záznam vidění kočky, nahoře originál a dole záznam[26]



Obrázek 9: Originál schématu systému BCI podle Miguela Nicolelise[10]

nich a používat ji jako by byla jejich vlastní. BCI používalo predikci rychlosti pohybu a síly úchopu.

4.2 Výzkum prováděný na lidech

Výzkum BCI systémů prováděný na lidech je, co se týče invazivního a semi-invazivního, zaměřen hlavně na obnovu smyslů jako je zrak či sluch nebo na alespoň částečné znovuzískání pohybových schopností postižené části po úrazu.

- **Obnovení zraku**: Jako jeden z prvních vědců, který se podílel na takovémto zařízení, byl William Dobelle. Ten se zaměřil na výzkum částečného obnovení zraku u jedinců s úplnou ztrátou zraku. První prototyp (1978) obsahoval 68 elektrod implantovaných do primární zrakové oblasti muže který v dospívání oslepl. Aparát se skládal z kamery na brýlích připojené na počítač. Tento BCI systém umožnil muži vidět odstíny šedi v nízkých hodnotách snímků za sekundu. Druhý prototyp

byl vytvořen v roce 2002 a bylo to jedno z prvních zařízení tohoto typu vyrobené pro komerční použití. První sadu těchto přístrojů si koupilo 16 pacientů. Tato generace zařízení byla mnohem skladnější a tudíž jej mohli pacienti používat i v každodenním životě. Zařízení umožňovalo pacientům vnímat základní obrazové jevy a prakticky ihned po zákroku byli schopni řídit auto na parkovišti ve velmi pomalé rychlosti.

- **Obnovení pohybu**: Další oblastí zaměření vědeckých týmů vyvíjející BCI systémy je obnovení pohybu paralyzovaných pacientů, nebo propojení jejich mozku s počítačem ovládající např. robotickou ruku [10]. V roce 2005 nastal průlom v tomto odvětví, kdy se Matt Nagle (kvadruplegik) stal první osobou ovládající robotickou ruku pomocí systému BCI. Tento systém vyvinul profesor Jonathan Wolpaw a jeho tým. Jejich zařízení obsahovalo 96 elektrod připojených na pravou část primární motorické oblasti umožňující Mattovi hýbat s robotickou rukou pouze myšlenkami. Toto zařízení využívalo semi-invazivní technologii.

- **Neinvazivní systémy**: Vedle použití invazivních metod se vyskytují také BCI systémy založené na neinvazivním přístupu k získání mozkové aktivity pracující na základě EEG. Tyto zařízení mají velkou výhodu v nenáročnosti použití avšak za cenu nízké kvality signálu a velkého šumu. Další nevýhodou je potřeba dlouhého tréningu a velkého soustředění k pozitivním výsledkům. Např. v 90 letech Niels Birbaumer vytvořil BCI systém založený na EEG, který svým pacientům umožnil pohyb kurzorem. Pacientům ale trvalo několik měsíců naučit se napsat pomocí tohoto kurzoru 100 znaků za hodinu. Pokrok nastal s využitím tzv. P300 vln, které jsou generovány spontánně když lidé vidí něco co znají. V 80. letech bylo sestrojeno zařízení, které pomocí těchto vln umožňovalo subjektům se syndromem uzamčení zadávat slova, písmena a jednoduché úkoly počítači a umožnilo jim také přes syntetizér mluvit. Na základě dalších výzkumů se dokazálo, že potenciál systémů využívající EEG se blíží potenciálu invazivních systémů. Např. v roce 2013 byl představen systém pro ovládání dálkově řízené kvadroptéry [11]. Vznikl i obor herního průmyslu, tzv. neurogaming, zabývající se využití těchto zařízení při hraní počítačových her. Na trhu také existuje několik dostupných zařízení, která lze použit pro zábavní účely jako např. NeuroSky nebo Emotiv Epoc, který je použit v této práci.

Obrázek 10: Pacientka ovládající robotickou ruku pomocí semi-invazivního BCI[29]

Obrázek 11: Subjekt ovládající kvadroptéru pomocí neinvazivního BCI[28]

5 Implementace

Hlavním cílem této práce bylo vytvořit demo scénu využívající zařízení s rozpoznáváním mozkové aktivity. Pro tento účel bylo vybráno, již dříve zmíněné, zařízení Emotiv EPOC. Úkolem bylo vizualizovat motorické stavy subjektu, pro které nebyl tento headset přímo zkonstruován, a tak zjistit jeho možnosti v těchto oblastech. Aplikace je však napsána tak, aby bylo možné připojit jakékoliv zařízení rozpoznávající mozkovou aktivitu. Náplní této práce bylo také vytvořit aplikaci pro správu headsetu a vše potřebné pro nahrávání a vytváření záznamů pro rozpoznávané akce. Tato aplikace je rovněž nezávislá na rozpoznávaném zařízení, ale i na demo scéně, pouze s limitací na počet elektrod a to na 128 (Limitace dána systémem 10/5). Aplikace využívá knihovnu pro rozpoznávání akcí od Martina Bartečka, se kterým jsem úzce spolupracoval.

5.1 Použité technologie

Obě aplikace jsou napsány v jazyce C++ s využitím grafického API OpenGL pro demo scénu a frameworku Qt v aplikaci pro správu headsetu. Tyto technologie bude popisovat následující podkapitola. Využití frameworku .NET bylo v průběhu vývoje nahrazeno Qt frameworkem, který byl vhodnější pro propojení s OpenGL aplikací využívající knihovnu FreeGlut pro správu oken.

5.1.1 OpenGL

Je standard specifikující multiplatformní rozhraní(API) pro tvorbu aplikací využívající 3D a 2D grafiku[8]. API OpenGL[13][14][15][16] existuje pro všechny počítačové platformy a je nezávislá na použitém hardwaru, avšak daný hardware podporuje tuto knihovnu jen do určité verze. OpenGL je volně dostupným produktem a je spravováno neziskovou organizací Khronos Group[12][12]. Současná verze je 4.5. OpenGL samotné se nestará a vytvoření okna pro vykreslení grafiky, a tak je zapotřebí využití dalšího nástroje. V této práci byla k tomuto účelu použita knihovna Glut, a to konkrétně v distribuci FreeGlut. Dále byla použita knihovna FreeImage pro načítaní obrazových dat pro 1D a 2D textury. OpenGL obsahuje dva profily, a to tzv. compatibility a core profile. Compatibility profile je určen pro zpětnou kompatibilitu se staršími verzemi karet využívající tzv. fixed function pipeline. Tento profil je však pro tuto práci nevhodný jelikož neposkytuje kýženou funkcionalitu. V následujících odrážkách jsou popsány hlavní vlastnosti OpenGL v režimu Core profile.

- Geometrie jsou v OpenGL reprezentovány pomocí bodů, úseček, trojúhelníků, trojúhelníkových pásů či čtyřúhelníků.
- Vykreslování je prováděno do Framebufferů, ke kterým je možné připojit různé renderbuffery či textury.
- OpenGL umožňuje ukládání dat pro geometrie, textury a jiná data do bufferů, které jsou uloženy přímo v paměti grafické karty.

Obrázek 12: Logo OpenGL a Khronos group[13][12]

- Umožňuje vykreslování částečně průhledných ploch pomocí mixování barev fragmentů či jejich alfa složek .
- Pro vykreslení jakékoliv geometrie je třeba definovat program obsahující tzv. shadery, které se starají o to, jak má být daná geometrie vykreslena. Shadery jsou následujícího typu.
  - *Vertex Shader*: je program který se provede pro každý vrchol geometrie, slouží většinou k transformaci vrcholu a k výpočtu informací pro fragment shader.
  - *Fragment Shader*: je program pracující s fragmenty povrchu geometrie po rasterizaci. Využívá se např. k nasvícení scény, nebo aplikaci materiálů či efektů.
  - *Geometry Shader*: slouží, stejně jako vertex shader, pro úpravu vrcholů geometrie, avšak s tím rozdílem, že může přidávat či odebírat vrcholy této geometrie.
  - *Compute Shader*: se využívá k urychlení algoritmů zpracovávajících ať už geometrii nebo jiná data. Výpočet, jako u všech shaderů, probíhá na grafické kartě.
  - *Tessellation Shader*: se skládá ze tří shaderů z nichž jsou dva programovatelné, a to tesselation control a tesellation evaluation. Tyto shadery umožňují měnit geometrii pomocí dělení primitiv.

5.1.2 Qt framework

Je multiplatformní knihovna pro implementaci programů s grafickým uživatelským prostředím [13]. Knihovna byla původně vytvořena společností Trolltech v roce 1999.[18] Poté ji společnost prodala firmě Nokia, v roce 2008, a nakonec ji odkoupila společnost Digia, která ji vlastní dodnes. Používá jazyk C++, ale existuje i pro jiné jazyky jako např. Python, C#, Java, Ruby a další. Standardní C++ však rozšiřuje o systém signálů a slotů značně ulehčujících implementaci událostí na základě změny stavu objektu. Tato knihovna má podporu pro práci se soubory, XML, SQL, grafikou či multimédii a je tak oblíbeným nástrojem pro tvorbu aplikací. Knihovna používá nativní vzhled operačního systému a tak se tyto aplikace vizuálně přizpůsobí do používaného prostředí. Výhodou této knihovny je také velice dobrá dokumentace na oficiálních stránkách výrobce. S knihovnou jsou distribuovány také vývojové prostředí QtCreator a QtDesigner. Lze nainstalovat

Obrázek 13: Logo Qt[17]

i plugin do Visual Studia. Aplikace vytvořené pomocí této knihovny je možno distribuovat pod licencí GPL, LGPL a za určitých podmínek mohou být distribuovány i komerčně. Knihovna je nyní ve verzi 5.4.

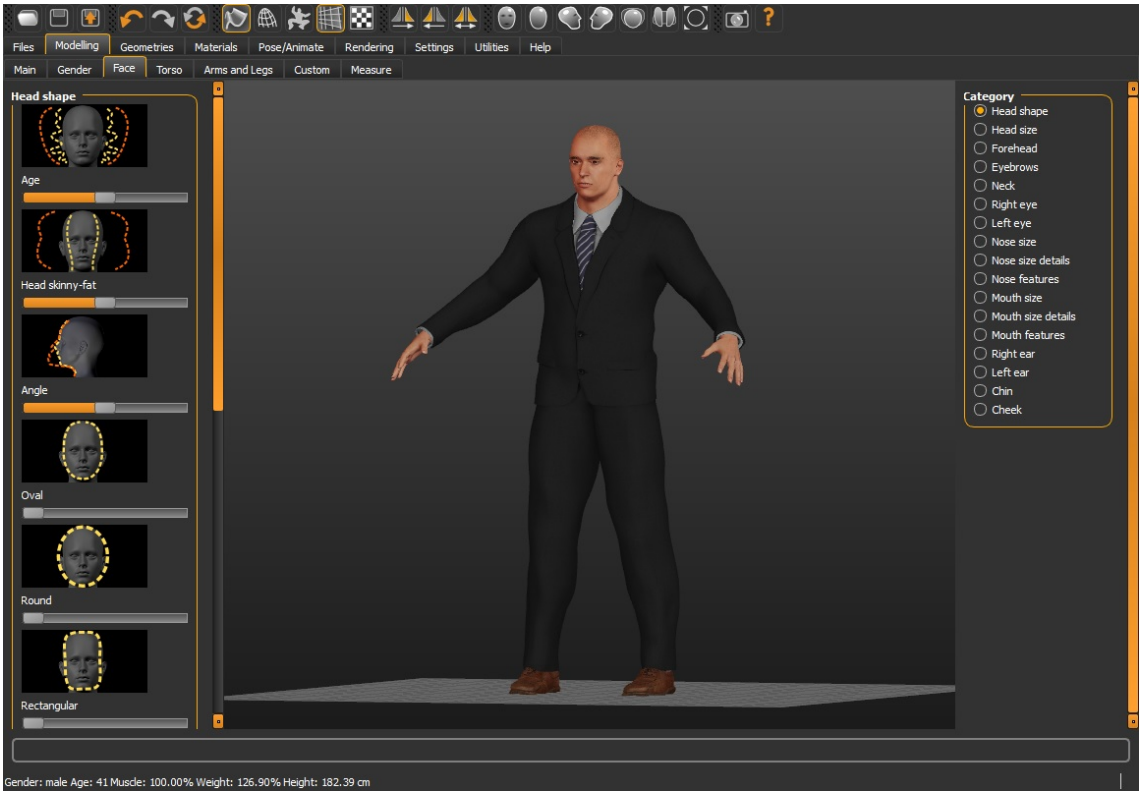
V této práci bylo nutné tento framework rozšířit o knihovnu QWT[19], která rozšiřuje Qt o widgety používané v technických aplikacích jako např. 2D Grafy, posuvníky, displaye, potenciometry a další. Knihovna byla použita kvůli grafům pro vizualizaci EEG signálu. Je zdarma a pod licencí QWT založenou na licenci LGPL.

5.1.3 MakeHuman

Je nástroj pro tvorbu anatomicky správných prototypů geometrií lidského těla. Je vyvíjen programátory, umělci a studenty se zájmem o 3D modelování.

Původ této aplikace sahá do roku 1999. Původně se jednalo pouze o skript jménem MakeHead do modelovacího nástroje *Blender*[21]. O rok později byl vytvořen tým a projekt dostal svůj nynější název, avšak stále byl napsán pouze pomocí jazyka *Python* s využitím API *Blenderu*. V roce 2005 byl kompletně přepsán do jazyka C a vydán jako samostatný nástroj. V dalších letech byl postupně přepisován do jádra napsaném v jazyce C++, které je doplněno o skripty v jazyce *Python*. Současná verze je 1.0.2.

Aplikace využívá technologii morfování geometrie. Základem je obecná geometrie lidské postavy, kterou lze upravit posuvníky do kýžené podoby. Na výběr je z mnoha atributů určující věk, pohlaví, rasu, výšku a další. Tuto postavu lze však dále upravovat pro jednotlivé části lidského těla. Lze tak např. nastavit proporce obličeje, rukou, nohou nebo jednotlivých prstů. Nástroj je také schopen přidat ke geometrii vlasy, obočí, oční bulvy, zuby nebo jazyk. K výsledné geometrii lze také nastavit kostru, která slouží k animaci postavy. Následně lze zvolit i z několika druhů oblečení a textur kůže, vlasů, obočí



Obrázek 14: Ukázka prostředí programu Makehuman[20]

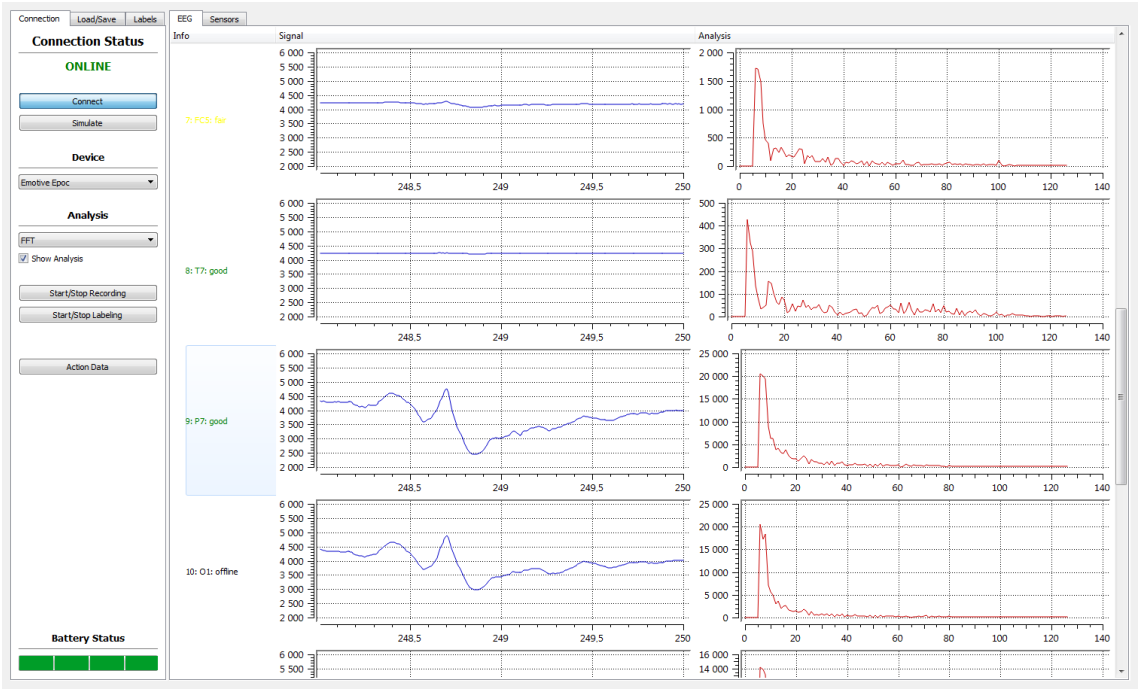
a očí. Výsledný model lze exportovat do několika standardizovaných formátů souboru. Aplikace je dále rozšířitelná o skripty psané v jazyce *Python* a obsahuje také nástroje pro program *Blender* spolu s výměnným formátem speciálně pro něj.

MakeHuman obsahuje také nastavení hustoty trojúhelníkové sítě. Pro účely aplikace byla zvolena varianta z velkou hustotou sítě kvůli malé vzdálenosti kamery od modelu. Byla tak vygenerována topologie s 91,552 vrcholy a 181,008 trojúhelníky.

Díky těmto vlastnostem je tento nástroj kompletním a nedocenitelným pomocníkem při tvorbě prototypů lidských postav využívaných k rekonstrukci vzhledu, při návrhu budov, industriálním designu nebo např. v herním průmyslu. Uživatelské prostředí lze vidět na obrázku[14]. MakeHuman je distribuován pod licencí AGPL a je zcela zdarma.

5.2 Aplikace pro správu headsetu

První z vytvořených aplikací je aplikace určena na správu headsetu. Uživatelské prostředí využívá frameworku Qt. Obsahuje nejdůležitější informace o aktuálním stavu zařízení, zobrazení signálu a spektrální analýzy, a vše potřebné k vytváření záznamů EEG signálu a vytváření jednotlivých akcí. Pro zachování univerzálnosti není tato aplikace přímo propojena s headsetem Emotiv EPOC. Obsahuje uživatelské prostředí pro ovládání headsetu, získávání dat z něj a logiku pro práci s těmito daty. Celá aplikace je roz-



Obrázek 15: Připojení a aktuální stav headsetu

dělena na dvě části umístěné vedle sebe, z nichž jedna obsahuje nastavení a druhá slouží k zobrazení a práci se signálem. Každá z nich je rozdělena na záložky. Tato aplikace je ve formě knihovny jménem *SignalGui* propojené s hlavní aplikací zobrazující demo scénu. Z Demo scény se také spouští a to tlačítkem EEG tools. V následujících podkapitolách jsou popsány jednotlivé části této aplikace.

5.2.1 Připojení a aktuální stav

První záložka nastavení obsahuje základní informace o stavu headsetu a některé ovládací prvky viz obrázek [15]. Pro připojení slouží tlačítko Connect. Poté, co se aplikace připojí, změní se červený nápis offline na online zelené barvy. Při výpadku spojení s headsetem se stav změní zpět na offline. Ve spodní části je widget typu progress bar zobrazující aktuální stav baterie. Jelikož je tato aplikace implementována s ohledem na univerzálnost, je zde ovládací prvek typu combobox pro výběr zařízení, i když byla použita pouze se zařízením Emotiv EPOC. Dále je zde volba typu spektrální analýzy a jestli se má zobrazovat vedle aktuálního signálu. Další možnou volbou je zapnutí nahrávání signálu a označování jeho částí za běhu. Tato funkcionalita, společně s zobrazením dat akcí pomocí tlačítka action data, je popsána v podkapitole 5.2.2.

Vedle sekce pro nastavení se nachází sekce skládající se ze dvou záložek. Záložka EEG zobrazuje data z jednotlivých senzorů ve formě EEG signálu. Počet senzorů se zadává v konstruktoru objektu *SignalGui*, který dědí z třídy *QMainWindow* reprezentující hlavní

obrazovku aplikace. V metodě *init()* se poté podle tohoto počtu inicializují objekty typu *SignalChannel*, které obsahují popisek a grafy pro každý kanál reprezentující EEG signál a spektrální analýzu. Tento objekt dědí z třídy *QTreeWidgetItem*, jelikož je prvkem v objektu typu *QTreeWidget*, který efektivně zobrazuje všechny tyto prvky pod sebou viz obrázek [15]. Popisek signálu každého senzoru obsahuje jeho název podle systémů 10/20 a aktuální stav kvality signálu obarvený příslušnou barvou. Data signálu senzoru jsou v grafu zobrazena změnou signálu závislou na čase. Grafy jsou objekty typu *QwtPlot*. Knihovna Qwt umožňuje do těchto grafů vkládat jeden či více objektů typu *QwtPlotCurve* reprezentující průběh veličiny zobrazený v grafu. Lze také zobrazit mřížku grafu a jednotlivým křivkám nastavit vzhled. Data se grafům přiřazují pomocí funkce *UpdateData*. Ta má v parametrech tři odkazy na pole hodnot. Dva jsou typu double a reprezentující data signálu(Osa Y) a čas (Osa X). Třetí je typu float a jsou v něm uložena data pro graf spektrální analýzy. Hodnot je vždy 256 pro data signálu a 128 pro spektrální analýzu. Přiřazení hodnot grafům probíhá pomocí funkce *setRawSamples()* objektu *QwtPlotCurve* daného grafu. Tato metoda přijímá parametry typu double pro osu X, osu Y a celočíselnou proměnou pro počet hodnot.

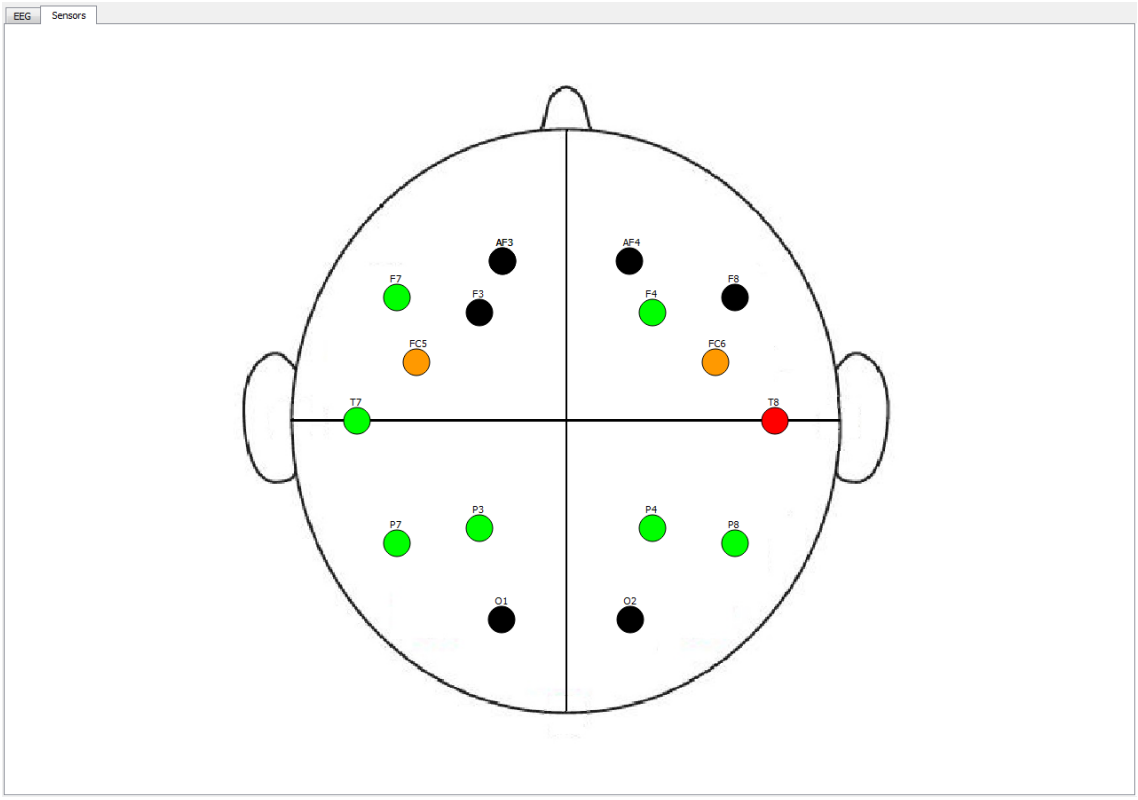
```
void initChannels(int count)
{
    noChannels = count;
    signalChannels.clear();

    actionsView = new ActionsView(noChannels);

    for(int i = 0;i<count;i++)
    {
        SignalChannel* channel = new SignalChannel(ui.signalTreeWidget);
        channel->initWidget(i + 1);

        connect(channel, SIGNAL(zoneSelected(QPointF)), this,
                SLOT(on_selectedSignalPlotZone(QPointF)));
        connect(channel, SIGNAL(rightClick(QPointF)), this,
                SLOT(on_rightClickedSignalPlotZone(QPointF)));
        connect(channel, SIGNAL(panning(int, int)), this,
                SLOT(on_plotPanned(int, int)));
        connect(channel, SIGNAL(plotZoomed(QRectF)), this,
                SLOT(on_signalPlotZoomed(QRectF)));
        connect(channel, SIGNAL(magnify(double)), this,
                SLOT(on_signalPlotMagnify(double)));

        signalChannels.push_back(channel);

        ui.signalTreeWidget->addTopLevelItem(channel);
        ui.signalTreeWidget->setItemWidget(channel, 0, channel->getOnlineLabel());
        ui.signalTreeWidget->setItemWidget(channel, 1, channel->getSignalPlot());
        ui.signalTreeWidget->setItemWidget(channel, 2, channel->getSpectrumPlot());
    }
```

```
        connect(actionsView, SIGNAL(on_selectedChannelAndAction(std::string,int)),this,
            SLOT(on_ActionChanged(std::string,int)));

        this->setCentralWidget(ui.centralWidget);
        plotZoneItemManager = new PlotZoneItemManager(count);
    }
```

Výpis 1: Inicializace seznamu senzorů a použití systému signálů a slotů

Druhá záložka, v téže sekci jako zobrazení EEG signálu, se nazývá Sensors viz obrázek [16]. Má na starosti zobrazení kvality snímaného signálu v jednotlivých senzorech. Senzory jsou zobrazeny jako obarvené body na siluetě lidské hlavy a nacházejí se na pozicích podle systému 10/20. O vykreslování senzorů se stará třída *SensorsGraphicsScene*, která dědí z třídy *QGraphicsScene*. Ta je přiřazena do instance widgetu *GraphicsView* vytvářející plochu pro jednotlivé grafické scény. Ve třídě *SensorsGraphicsScene* je inicializační metoda *initSensor*, která obsahuje dva parametry. Počet senzorů a kolekci textových řetězců reprezentující názvy všech senzorů headsetu. V této metodě se inicializuje velikost bodů reprezentující senzory a načtou se ze souboru předem vypočtené pozice senzorů na obrázku. Během načítání se ukládají pouze ty, které jsou předem určené v dříve zmíněné kolekci. Pozice jsou podle systému 10/20 a jsou uloženy pod standardizovanými názvy tohoto systému. Jsou to čísla typu float v rozmezí -1 až 1, a lze je tedy snadno přepočítat pro jakýkoliv obrázek reprezentující pozadí. V souboru je uloženo 128 pozic, a tak je možné ho použít pro jakýkoliv headset využívající systém 10/20. Jednotlivé senzory jsou instance třídy *SensorGraphicsItem*, která dědí z *QGraphicsItem*. Objekty tohoto typu lze přidávat do grafické scény a ty jsou pak vykreslovány na svých pozicích v ní. Obsahují metodu *paint()*, která se volá při každém vykreslení objektu a v ní je také definováno jak a co se bude vykreslovat. Vyobrazený senzor může mít následující barvy na základě kvality signálu.

- **Černá**: Bez signálu (Headset vypnutý nebo je špatně umístěný senzor)
- **Červená**: Velmi slabý signál
- **Oranžová**: Slabý signál
- **Žlutá**: Dobrý signál
- **Zelená**: Vynikající signál

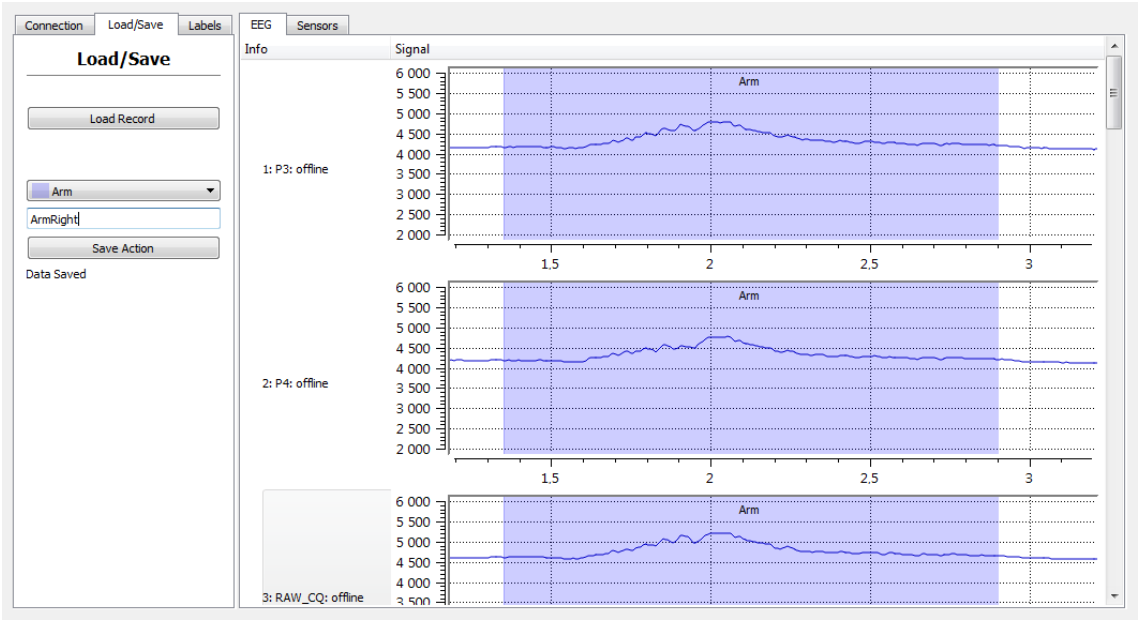


Obrázek 16: Zobrazení kvality signálu senzorů v rozmístění podle systému 10/20

5.2.2 Nahrávání,načítání záznamů a ukládání akcí

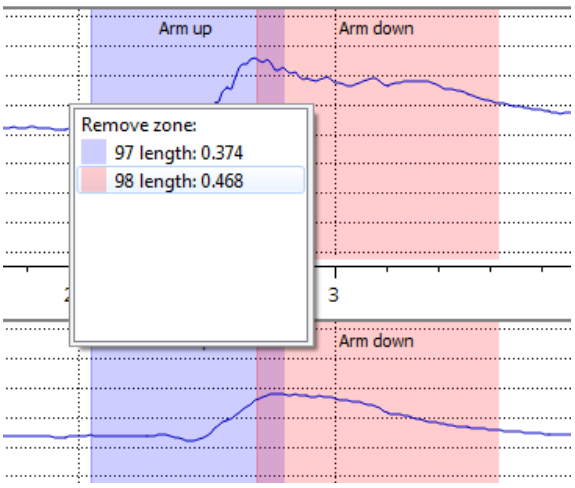
Nahrávání záznamů se zapíná pomocí tlačítka *Start/Stop Recording* umístěném na záložce viz obrázek Connection[15] v levé sekci aplikace. Toto tlačítko je třístavové a při spuštění nahrávání zůstane stisknuto tak, aby signalizovalo probíhající nahrávání. Během nahrávání lze také použít okamžité označování úseku signálu, a to tlačítkem *Start/Stop Labeling*, které je také třístavové. Po stisknutí tohoto tlačítka se začne označovat vždy časovou oblast začátku grafu a po opětovném stisknutí ukončí označovaní rovněž na časovém údaji na začátku grafu. Pro ukončení nahrávání je zapotřebí znova stisknout tlačítko *Start/Stop Recording*, které otevře dialog s polem pro zadání názvu souboru. Po stisknutí tlačítka *Save* se záznam uloží do složky *../Data/Records* pod daným názvem a příponou *.dataEEG*. Záznamy oblastí se ukládají do stejné složky a pod stejným názvem, ale příponou *.dataZones*. Pro načtení daného záznamu je třeba odpojit headset pomocí třístavového tlačítka *Connect* v levé sekci na záložce *Connection*. Poté je možno načíst uložený soubor pomocí tlačítka *Load Record* ve stejné sekci pod záložkou *Load/Save*. Data se ihned načtou i s označenými oblastmi.

Po načtení záznamu můžeme označit určité úseky, které je možno uložit jako akci, která bude poté rozpoznávána. K tomuto účelu slouží štítky, které lze spravovat v zá-



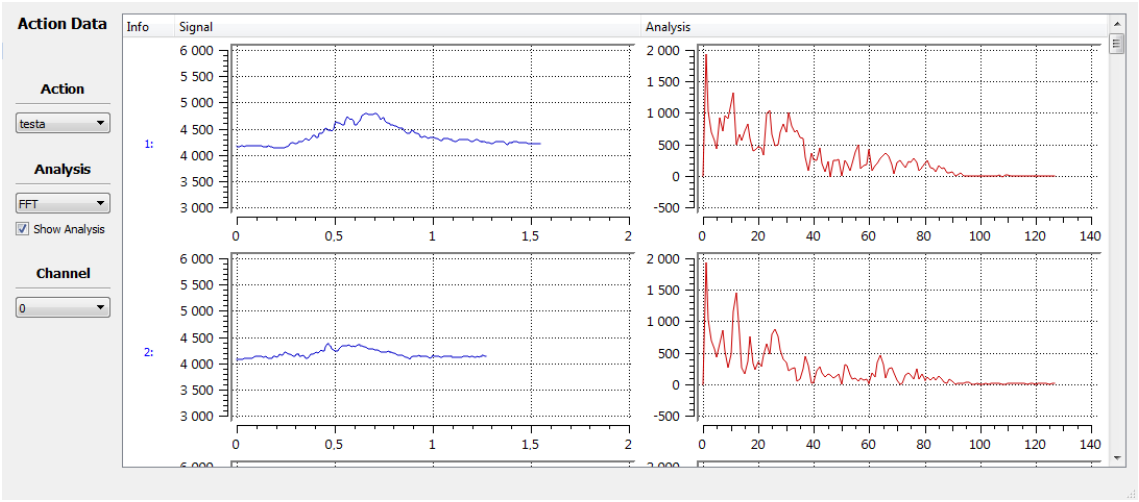
Obrázek 17: Načtení záznamu a uložení vybrané akce

Obrázek 18: Správa štítků a označení oblastí v grafu



Obrázek 19: Detail vymazávání štítků v signálu při překrytí

ložce *Labels*. Každý štítek má svoji barvu a název. Štítek lze vytvořit v části oddělené nápisem *Add Label* viz obrázek [18]. Nachází se zde textové pole pro zadání názvu štítku a combobox pro výběr z několika barev. Po stisknutí tlačítka *Add Label* se přidá vytvořený štítek do seznamu štítků. Ve vrchní části stejné záložky jsou, ve widgetu typu combobox, na výběr všechny vytvořené štítky. Pomocí tlačítka *Remove* lze vybraný štítek vymazat. Pro tuto funkcionalitu byla vytvořena třída *LabelManager*, která se stará o ukládání a načítaní štítků ze souboru a nastavování stavu ovládacích prvků. Každý nově přidaný štítek je uložen do souboru, vložen na první pozici v comboboxu a je nastaven jako výchozí. Načítání všech štítků probíhá v konstruktoru manageru. Tyto štítky lze použít pro označování určitých oblastí v načteném záznamu EEG signálu. Oblasti lze označovat přímo v grafech a to pouze stisknutím levého tlačítka myši na ploše grafu, posunutím kurzoru a puštěním tlačítka nad tou částí grafu, kde se má nacházet konec označené oblasti. Daná oblast se označí vybraným štítkem, a to jeho barvou a názvem viz obrázek [18]. Označování v grafu je realizováno pomocí instance třídy *PlotPicker*, která dědí z třídy *QwtPlotPicker* knihovny QWT. Třída *PlotPicker* obsahuje dva objekty typu *QwtPlotMarker* reprezentující hranice označované oblasti a instanci třídy *QwtPlotZoneItem* reprezentující oblast mezi nimi. O označení se stará metoda *move()*, která posouvá hranice označené zóny a vybarvuje vnitřek zóny barvou. Po vybrání oblasti je emitován signál *zoneSelected*. Na tento signál reaguje instance třídy *PlotZoneItemManager*. Tato třída má na starosti práci se zónami označenými v grafech. Stará se o přidávání a mazání zón z grafů, a také načítání a ukládání všech zón pro daný graf do souboru s příponou *.dataZones*. Při přidání zóny do tohoto manažeru se daná zóna zobrazí ve všech grafech, a to označena právě vybraným štítkem. Označenou oblast lze z grafu také vymazat, a to kliknutím pravého tlačítka myši na danou oblast. Pokud se v dané oblasti nachází více překrývajících se zón, zobrazí se nabídka s výběrem všech zón v daném místě. Obsahuje vždy index zóny, jeho barvu a délku viz obrázek [19]. Po vybrání je zóna vymazána ze všech grafů. Pro přehlednou práci se záznamem signálu jsou vždy zobrazeny dvě sekundy záznamu. Záznam lze



Obrázek 20: Zobrazení vzorků nahrané akce a jejích spektrální analýza

v grafu posouvat pomocí tzv. gesta swipe, implementovaným stisknutím klávesy shift, stisknutím levého tlačítka myši nad oblastí grafu a potažením do opačného směru než kam jej chceme posunout. Všechny grafy se následně posunou do stejné časové oblasti. Tato funkcionalita je zajištěna použitím instance třídy typu *QwtPlotPanner* a její přiřazení ke grafu. Záznamy lze rovněž přibližovat a oddalovat pro zobrazení detailu signálu. To je implementováno dvěma způsoby. První je implementován pomocí třídy *QwtPlotZoomer*, a to stisknutím klávesy Ctrl a označení obdélníkové oblasti tažení myši se stisknutým levým tlačítkem. Pro zobrazení původní oblasti lze stisknout klávesu Esc. Druhý způsob je díky třídě *QwtPlotMagnifier*, a to opět stiskem klávesy Ctrl a použitím kolečka myši přibližovat či oddalovat záznam. Daná oblast se v obou případech přiblíží či oddálí na všech grafech.

Pro uložení označených oblastí a zároveň definování a uložení dat akce slouží tlačítko *Save Action* na záložce *Load/Save*. Zóny se uloží pod jménem záznamu, avšak jimi definovaná akce se uloží pod jménem napsaným v textovém poli nad tímto tlačítkem. Uloží se rovněž pouze záznamy označené štítkem vybraným z widgetu typu combobox nad textovým polem pro název akce. V tomto comboboxu se nacházejí všechny typy štítků použitých v záznamu.

K zobrazení dat pro jednotlivé akce slouží tlačítko *Action Data* na záložce *Connection* viz obrázek [15]. Po stisknutí se otevře okno s grafy zobrazující signál jednotlivých vzorků viz obrázek [20]. V comboboxu pod nadpisem Action jsou na výběr jednotlivé akce. Stejně jako na záložce *Connection* v hlavním okně pro správu headsetu je i zde volba typu spektrální analýzy a to jestli se má zobrazovat, i když zde byla použita pouze furierova transformace. Tento combobox je zde opět kvůli univerzálnosti aplikace. Poslední možností je výběr senzoru, pro který se mají vzorky ukázat.

5.3 Demo scéna

Hlavní náplní této práce bylo vytvořit aplikaci pro vizualizaci motorických stavů lidského těla, snímaných EEG headsetem. Aplikace je vytvořena pomocí programovacího jazyka C++, knihovny OpenGL pro vykreslování 3D scény a knihovny AntTweakBar pro tvorbu jednoduchého uživatelského prostředí. Tato aplikace také obsahuje propojení mezi knihovnou pro získávání dat a rozpoznávání akcí z headsetu a aplikací pro jeho správu. Je tak hlavní aplikací řídící chod celého tohoto systému.

O toto propojení se stará třída *GUItoEEGmanager* obsahující metodu *Update()*, která následně volá metody pro aktualizaci stavů propojených komponent roztříděných podle jejich účelu. Tato metoda je volána ve speciálním vlákně a to třicetkrát během vteřiny. Tímto je omezena výkonová zátěž tohoto procesu. Tato třída rovněž obsahuje vše potřebné k tomuto propojení včetně instancí obou komponent. Odděluje tak tuto logiku od zbytku logiky aplikace demo scény, a ta je tudíž nezávislá na obou těchto komponentách.

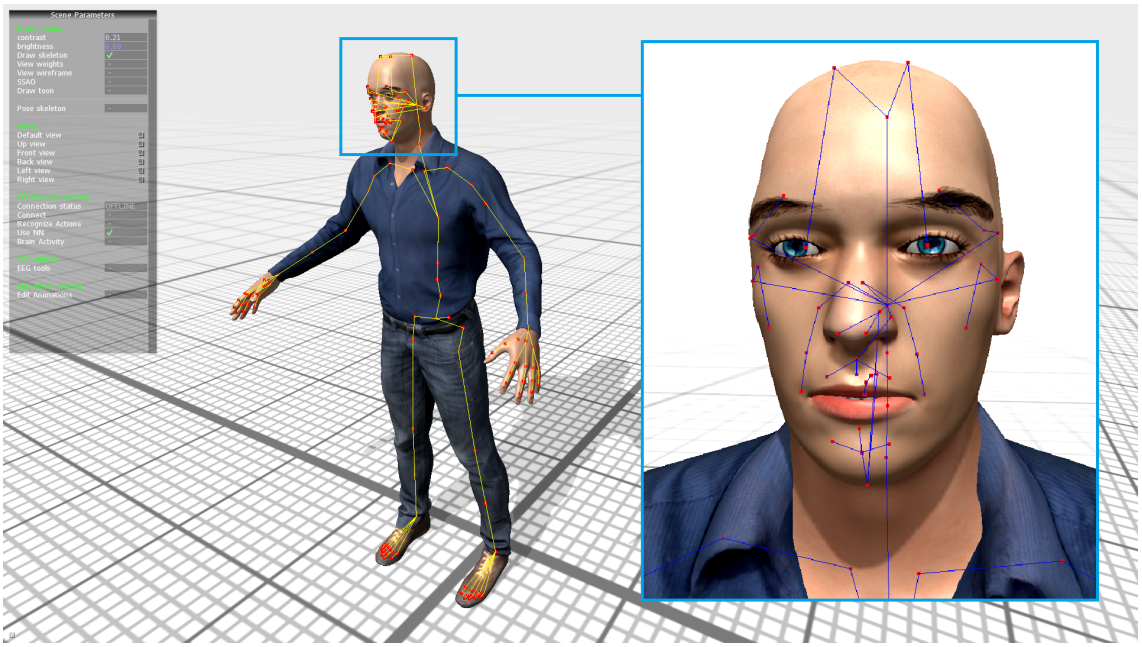
5.3.1 Uživatelské prostředí

K ovládání aplikace slouží lišta s ovládacími prvky umístěná na levém okraji obrazovky. Nachází se zde několik sekcí pro ovládání dané části aplikace. První sekce s názvem *Draw options* má na starosti změnu vizualizace scény. Nachází se zde zapnutí/vypnutí vykreslování kostry modelu, vizualizace vah kostí, zobrazení drátového modelu, SSAO a nefotorealistické zobrazení modelu. Mimo to zde lze nastavit jas a kontrast obrazu. V další sekci lze zapnout ve scéně možnost pohybovat s kostmi kostry modelu pomocí tlačítka *Pose skeleton*. Sekce pod ní slouží k rychlému výběru pohledu kamery. Na výběr je z pěti směrů kolem modelu plus pohled z pravého horního rohu nastavený jako základní. Další sekce je věnována rychlému zapnutí headsetu, výpisu jeho stavu, zapnutí rozpoznávání akcí, výběru rozpoznávací metody a zapnutí vizualizace mozkové aktivity. V předposlední sekci je zapnutí výše zmíněné aplikace pro správu headsetu, záznamů a akcí. Poslední sekce slouží k vytváření animací k definovaným akcím.

Uživatelské rozhraní je definováno v třídě *GUIManager*. Zde je v metodě *init()* vytvořena samotná lišta, a to pomocí funkce *TwDefine()*. K nim jsou přiřazovány ovládací prvky typu tlačítko pomocí funkcí *TwAddButton()*. Zatrhávající políčka jsou přidávány funkcí *TwAddVarRW()*. Všem těmto ovládacím prvkům je třeba nastavit popisek, štítek do kterého se mají vykreslovat a proměnnou či funkci, která bude obsluhovat změnu stavu. Třída *GUIManager* obsahuje i definice štítků pro uživatelské rozhraní editace animací akcí, které bude popsáno v podkapitole [5.3.3]. Je typu singleton proto, aby byla přístupná v jakékoliv části programu aplikace.

5.3.2 Kosterní systém

Hlavním úkolem aplikace je vizualizace motorických stavů lidské postavy v 3D prostoru. K vizualizaci tak nestačí pouze 3D model lidské postavy. Je zapotřebí tento model plynule deformovat do polohy reprezentující rozpoznaný motorický stav lidského těla. Jednou z možností jak toho dosáhnout je použití kosterního systému, který se svou variabi-



Obrázek 21: Kostra modelu, vpravo zobrazen detail obličeje

litou nejlépe hodí pro tuto aplikaci. Pro tento účel byl zvolen formát souboru *MD5* , který může obsahovat jak definici geometrie tak i kosterní systém a díky své struktuře jej lze snadno načíst. Model byl vytvořen pomocí programu *MakeHuman* a převeden do *MD5* formátu za pomocí modelovacího programu *Blender*.

Základem kosterního systému jsou kosti a váhy, které jsou přiřazeny jednotlivým vrcholům geometrie. Kosterní systém je ve většině případů tvořen stromovým systémem, kdy každá kost má svého rodiče kromě kosti reprezentující kořen stromu. Každá kost je definována svou pozicí a orientací. Ve formátu *MD5* je kosterní systém definován klouby, avšak princip je totožný s reprezentací pomocí kostí. Orientace kloubu je reprezentována quaternionem závislým na svém rodiči. Je tudíž v lokálním prostoru daného kloubu. Pozice je v globálním prostoru modelu. Ke každému vrcholu geometrie je definována sada indexů jednotlivých kloubů a sada vah, v rozmezí 0 až 1, které určují v jakém poměru daný kloub ovlivňuje pozici vrcholu.

Formát *MD5* definuje váhy, vrcholy, trojúhelníky a kosterní systém v souboru s příponou *.md5mesh*. Všechny tyto prvky jsou definovány zvlášť, avšak vždy s referencí na ostatní prvky tak, aby šlo sestavit geometrii propojenou s příslušným kostrou. Klouby jsou uloženy pomocí jména, pozice, orientace a indexu rodiče ve stromové struktuře. Váhy jsou reprezentovány indexem, indexem kloubu, samotnou vahou a pozicí v 3D prostoru a jsou uloženy za sebou pro jednotlivé vrcholy. Vrcholy jsou definovány souřadnicemi na textuře, počáteční vahou a počtem vah. Pozice vrcholů je třeba vypočítat z pozic kloubů a vah ovlivněných orientací kloubů a hodnotami jednotlivých vah. Rovněž normály vrcholů je zapotřebí vypočítat. Vrcholy jsou spojeny trojúhelníky, které jsou

definovány indexy jednotlivých vrcholů. Formát *MD5* obsahuje rovněž soubor s příponou *.md5anim*, který slouží k přenosu již předem vytvořené animace. Jelikož však aplikace obsahuje vlastní editor animací, je použití těchto souborů zbytečné. Tento formát rovněž obsahuje u každé části geometrie seznam názvů textur, případně způsob jejich kombinace. To je však pro definici materiálu dané části geometrie nedostatečné, a tak byl daný systém rozšířen o materiálový soubor formátu *.mtl* obsahující kýžené informace. Název tohoto souboru je v hlavičce souboru *.md5mesh* a u každé části geometrie je název příslušného materiálu. Tento formát je rovněž rozšířen o soubor definující maximální a minimální úhly jednotlivých kloubů ve všech osách. Tento soubor je poté při načítání modelu načten a jednotlivé hodnoty jsou přiřazeny daných kloubům.

Model a jeho kosterní systém reprezentuje třída *RiggedModel*. Ta obsahuje metody pro načtení *MD5* modelu, výpočet pozic vrcholů a výpočet normál. Obsahuje rovněž metody pro přepočet pozic a orientací kloubů při vytváření animací modelu. V následujících odrážkách jsou popsány hlavní metody této třídy.

- **LoadModel()**: Má na starosti načtení informací o modelu a jeho kosterním systému do předem vytvořených struktur reprezentující vrchol, trojúhelník, kloub a váhu. Pole těchto struktur jsou uložena v objektu třídy *Mesh* reprezentující část modelu. Pole těchto části pak tvoří celý model. Po načtení každé části modelu jsou volány metody *PrepareMesh()* a *PrepareNormals()* pro vypočítání pozic vrcholů a jejich normál. Po načtení celého modelu je zavolána metoda *CreateBoneMatricesFromJoints()* pro vytvoření matic kloubů potřebných k deformaci geometrie v shaderech.
- **PrepareMesh()**: Obsahuje výpočet pozic vrcholů na základě načtených pozic vah a kloubu a orientace těchto kloubů. Ukázka kódu výpočtu těchto souřadnic je uvedena níže.
- **PrepareNormals()**: Slouží k výpočtu vyhlazených normál geometrie. Prochází trojúhelníky geometrie a vypočítává normálu na základě vektorového součtu dvou směrových vektorů definující trojúhelník. Tyto normály pak přičítá k, již v předchozích krocích již vytvořeným, normálám geometrie a tvoří tak vyhlazené normály po celém povrchu.
- **rotateCurrentJoint()**: Je vytvořena pro rotaci zvoleného kloubu o určitý počet stupňů. Každý kloub obsahuje hodnoty pro aktuální, maximální a minimální úhel rotace v každém směru v lokálním prostoru daného kloubu. Na začátku této metody se kontrolují hranice rotací a případně přičítají aktuální úhly. Rotovat je zapotřebí i potomky daného kloubu ve stromové struktuře. Jelikož každý kloub obsahuje pouze index svého rodiče, je zapotřebí najít všechny potomky a rotovat je podle vybraného kloubu. K tomuto účelu slouží metoda *rotateChildJoints()*, která rotuje pozici a směr všech potomků kloubu vůči jeho pozici a směru. Na konci této metody je volána metoda *CreateBoneMatricesFromJoints()* pro přepočet transformačních matic.
- **CreateBoneMatricesFromJoints()**: Vytvoří z pozic a směrů všech kloubů matice pro přepočet pozic vrcholů v shaderech. Tento princip je vysvětlen v následujícím odstavci.

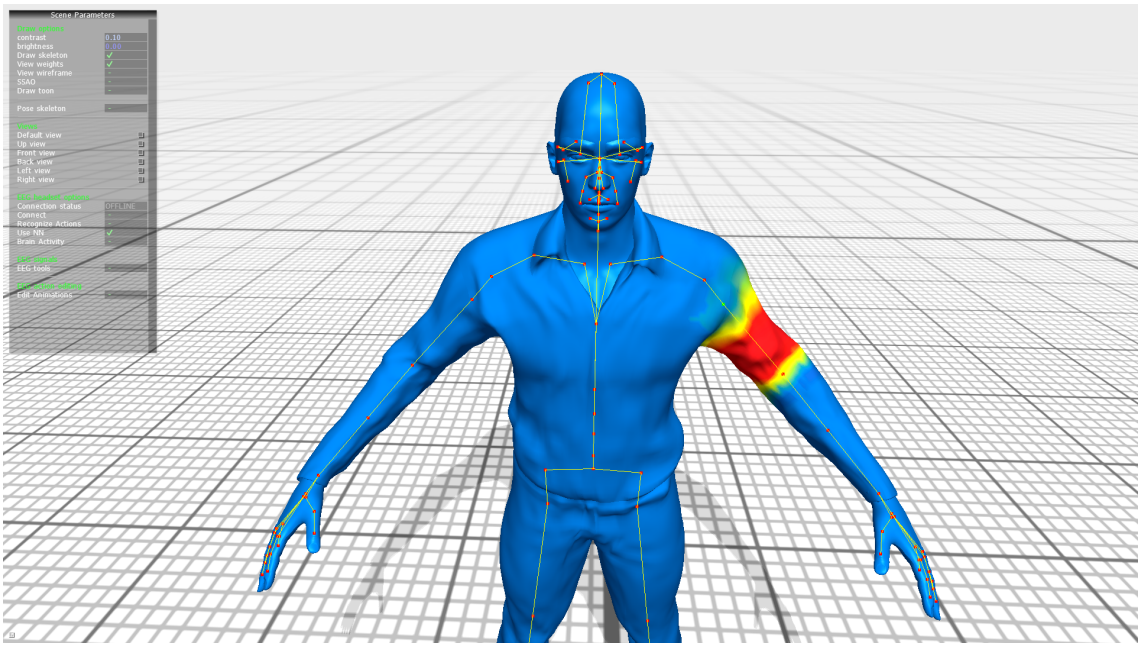
Pro změnu tvaru geometrie, podle pohybu kloubů kostry, je potřeba změnit pozici všech jejich vrcholů. Jelikož je tato operace výpočetně náročná, je změna pozic a orientací kloubů převáděna do transformačních matic, které jsou následně použity v shaderu pro přepočet pozic samotných vrcholů. Nejnáročnější operace je tak přemístěna na grafickou kartu a je tak urychlen její výpočet. Do shaderů se tak posílají nejen informace o pozicích vrcholů, normály a souřadnice vrcholů na textuře, ale i indexy v poli transformačních matic a váhy jejich vlivu. Indexy i váhy jsou reprezentovány vektorem s dimenzí o velikosti 4. Jeden vrchol tak mohou ovlivňovat maximálně čtyři klouby. Vždy se tak v shaderu sčítají 4 matice, každá vynásobena váhou. Výsledkem je matice, kterou se vynásobí vrchol a se tak změní jeho aktuální poloha v prostoru.

```
for (unsigned int i = 0; i < mesh.vertices.size(); i++)
{
    Vertex& vert = mesh.vertices[i];

    for (unsigned int j = 0; j < vert.m_WeightCount; j++)
    {
        Weight& weight = mesh.weights[vert.m_StartWeight + j];
        Joint& joint = joints[weight.jointId];

        glm::vec3 rotPos = joint.orientation * weight.position;

        vert.position += (joint.position + rotPos) * weight.bias;

        if (j < 4)
        {
            vert.boneIndices[j] = (float)weight.jointId;
            vert.boneWeights[j] = weight.bias;
        }
    }

    mesh.positions.push_back(vert.position);
    mesh.tCoords.push_back(vert.tcoord);
    mesh.boneIndexes.push_back(vert.boneIndices);
    mesh.boneWeights.push_back(vert.boneWeights);
}
```

Výpis 2: Ukázka výpočtu souřadnic vrcholů na základě údajů obsažených v .md5mesh souboru

5.3.3 Vytváření a editace animací

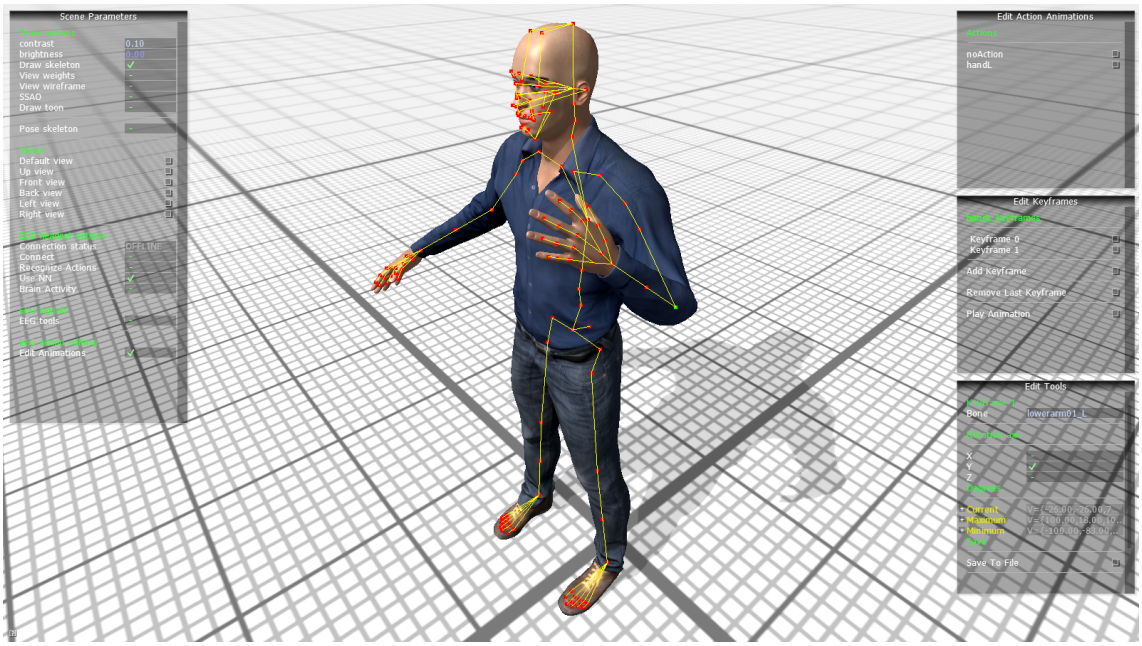
Aby vytvořená aplikace demo scény splňovala vše potřebné k vizualizaci motorických stavů lidského těla, bylo zapotřebí vytvořit nástroj pro vytváření animací, které představují právě tyto stavy. Tento nástroj je implementován přímo do demo scény. Spouští se kliknutím na tlačítko *Edit Animations*. Po kliknutí se v pravém horním rohu zobrazí lišta se seznamem všech akcí, vytvořených v aplikaci pro správu headsetu. Po kliknutí na danou akci se zobrazí další lišta se seznamem klíčových snímků dané aplikace. Klí-



Obrázek 22: Vizualizace vah kloubů kosterního systému

čové snímky reprezentují stav geometrie. Mezi klíčovými snímky se provádí lineární interpolace tak, aby vznikl spojitý vjem pohybu geometrie reprezentující lidskou postavu. Přidávání klíčového snímku se provádí tlačítkem *Add Keyframe* a lze také vymazat poslední vytvořený klíčový snímek pomocí tlačítka *Remove Last Keyframe*. Po přidání klíčového snímku se zkopíruje stav všech kloubů geometrie do následujícího snímku tak, aby šlo navazovat plynule ve změně stavu geometrie v dalším snímku. Poslední tlačítko na této liště je *Play Animation*, které přehraje animaci. Pro změnu stavu geometrie v daném klíčovém snímku je zapotřebí kliknout na vytvořený klíčový snímek. Otevře se lišta s nastavením. Poté lze kliknutím označit kloub, který chceme rotovat. O výběr se stará třída *Raytracer*, která pomocí stejnojmenného algoritmu hledá nejbližší kloub. Pomocí stisknutí klávesy R a pohybu myši ve vertikálním směru lze s tímto kloubem rotovat. V tomto režimu lze stisknout klávesu X,Y nebo Z pro výběr osy, ve které bude prováděna rotace. Opětovným stisknutím klávesy R se zruší možnost rotování a kloub zůstane v daném stavu. Lišta zobrazená po vybrání klíčového snímku obsahuje informace o vybraném kloubu, kterým chceme rotovat. Jedná se o jméno kloubu, jeho aktuální, minimální a maximální úhly rotace a vybranou osu rotace. Ve spodní části se nachází tlačítko *Save To File*, které slouží k uložení animace do souboru.

O správu těchto animací se stará třída *EEGActionManager*. Tato třída obsahuje kolekci objektů třídy *EEGAction* reprezentující animace přiřazené k předem vytvořeným akcím. Třída *EEGAction* obsahuje kolekci tříd *Keyframe* obsahující pozice a orientace všech kloubů modelu v daném klíčovém snímku. Třída *EEGAction* rovněž obsahuje metody pro načítání a uložení všech svých klíčových snímků do souboru s příponou *.jointInfo* do



Obrázek 23: Uživatelské rozhraní pro tvorbu animací

stejné složky, kde jsou uloženy vzorky všech uložených akcí. Všechny klíčové snímky dané akce jsou uloženy do jednoho souboru reprezentujícího danou akci. Jsou zde uloženy všechny pozice a orientace kloubů v daném klíčovém snímku s tím, že klíčové snímky jsou odděleny složenými závorkami. Třída obsahuje rovněž proměnnou, která určuje jak dlouho daná animace bude trvat. Ta je poté nastavena při rozpoznání akce, a to na základě amplitudy signálu rozpoznané akce.

Přehraní animace, po rozpoznání akce, má na starosti metoda *performActionEEG()* ve třídě *Entity_Animated*. Entitní systém reprezentace objektů ve scéně bude popsán v podkapitole [5.3.4]. Metoda *performActionEEG()* pouze nastaví v této třídě proměnnou *currentAction* na instanci třídy *EEGAction* reprezentující akci k provedení. V třídě *Entity_Animated* je metoda *update()*, která se provádí v předem daných časových intervalech. Obsahuje interpolaci pozic a orientací všech kloubů mezi jednotlivými klíčovými snímky v závislosti na čase.

5.3.4 Vizualizace scény

Pro splnění hlavního cíle této aplikace byla zvolena vizualizace motorických stavů ve 3D scéně. Byl kladen důraz na jednoduchost scény zaměřené na vizualizaci lidské postavy a její animace. Pomocí nastavení popsaného v kapitole [5.3.1] lze měnit charakter vzhledu scény. V následujících odstavcích a podkapitolách jsou popsány hlavní prvky vizualizace a jejich implementace.

Objekty ve scéně jsou reprezentovány třídami, které dědí z obecné třídy *Entity*, obsahující deklaraci metod a proměnných společných pro každý objekt ve scéně. Jsou to

především metody *draw()*, *drawForShadows()*, *update()* a *init()*, díky nímž lze přistupovat stejným způsobem k objektům s různými vlastnostmi. Metoda *init()* obsahuje inicializaci geometrie a atributů objektu, metoda *draw()* se stará o vykreslení objektu a metoda *update()* pro aktualizaci stavu objektu či jeho animace. Funkce *drawForShadows()* obsahuje vykreslení geometrie pro zápis do hloubkové textury použité k vytvoření stínu geometrie. V každém tomto derivátu třídy *Entity* je uložen objekt typu *Model*(Obj formát) nebo *RiggedModel* (formát MD5), ve kterém jsou uloženy informace o geometrii a metody pro vytvoření bufferů, které jsou v metodě *init()* načteny do paměti grafické karty. Všechny vytvořené *entity* jsou uloženy do kolekce ve třídě *SceneManager*, která je hlavní třídou celé aplikace. Pro účely této aplikace byly implementovány třídy *Entity_Animated*, *Entity_Plane*, *Entity_Human* a *Entity_Brain* reprezentující animovanou postavu, jednoduché okolní prostředí, statickou lidskou postavu a mozek pro vizualizaci mozkové aktivity. V následujících odstavcích si popíšeme vizualizaci provedenou v těchto třídách. Vizualizace mozkové aktivity je popsána v podkapitole [5.3.5].

Třída *Entity_Plane* obsahuje načtený model čtverce, který je zvětšen aby přesahoval rozmezí dohledu kamery a reprezentuje tak podstavu pro model lidské postavy. Na tento model je opakovaně nanesena textura jemné mřížky, která má definované mipmapy tak, aby se ve vzdálenosti od kamery ztrácel její vzor. Tato plocha je vykreslována bez požití osvětlovacího modelu , pouze je zde zobrazen stín postavy, která se na ní nachází.

Další výše zmíněnou třídou je třída *Entity_Animated*, která se stará o vykreslení a správu animovaného modelu lidské postavy. Tato třída obsahuje několik režimů vykreslování, které lze přepínat na hlavní liště uživatelského prostředí. Jednotlivé režimy, reprezentované metodami používající různé shadery, jsou popsány v následujících odrážkách.

- **DrawBasic()**: Tato metoda vykresluje lidskou postavu nasvícenou bodovým světlem. Pro nasvícení geometrie lze použít několik metod, v práci byla použita metoda využívající Phongova osvětlovacího modelu. Ten je založen na výpočtu ambientní, difúzní a zrdcadlové složky. Difúzní složka je vypočtena pomocí kosinu úhlu mezi normálou a směrem paprsku světla k fragmentu. Zrcadlová složka je pak vypočítána na základě kosinu úhlu mezi odraženým paprskem světla od fragmentu a směrem pohledu z kamery. Zrcadlová složka je následně umocněna koeficientem svítivosti materiálu lidské postavy. Na postavě jsou vykresleny také stíny. Pro výpočet stínů v počítačové lze grafice použít několik metod. Jednou z nich je vykreslení ostrých stínů za použití hloubkových map. Tyto stíny lze poté filtrovat a získat tak měkké stíny. Pro výpočet těchto stínů se používá dvou průchodová metoda. Při prvním průchodu je generována hloubková mapa, obsahující vzdálenost jednotlivých fragmentů scény z pohledu světla. V druhém průchodu se vykreslí scéna z pohledu kamery. Pozice objektů se převede pomocí transformační matice do prostoru světla, zjistí se pozice na hloubkové mapě a z ní se získá vzdálenost. Pokud je vzdálenost fragmentu větší než vzdálenost uložena v hloubkové mapě, fragment je ve stínu, pokud není, tak je osvětlený. Pro zjemnění stínů byla použita technika *PCF*, která funguje na principu zprůměrování hodnot příspěvku okolních fragmentů ke stínu. Byla vyžita částečná hardwarová podpora této metody výrobcem grafických karet *NVIDIA*, a to nastavením parametru *GL_TEXTURE_COMPARE_MODE*

hloubkové textury na *GL_COMPARE_R_TO_TEXTURE* a použití sampleru typu *sampler2DShadow* ve fragment shaderu.

- **DrawWireframe()**: Je metoda zobrazující trojúhelníkovou síť modelu, která je vykreslena na model postavy, na kterém je nanesena pouze difúzní textura viz obrázek [24]. Tento režim umožňuje prohlédnout si vygenerovanou geometrii aplikací MakeHuman, a lze se tak přesvědčit o kvalitách tohoto programu. Trojúhelníková síť je pravidelná a lze vidět její zhuštění, např. v místech obličeje, která jsou obecně problematická při použití kosterního systému k deformaci geometrie modelu. Vykreslování drátového modelu je nastaveno pomocí funkce *glPolygonMode(GL_FRONT_AND_BACK, GL_LINE)*.

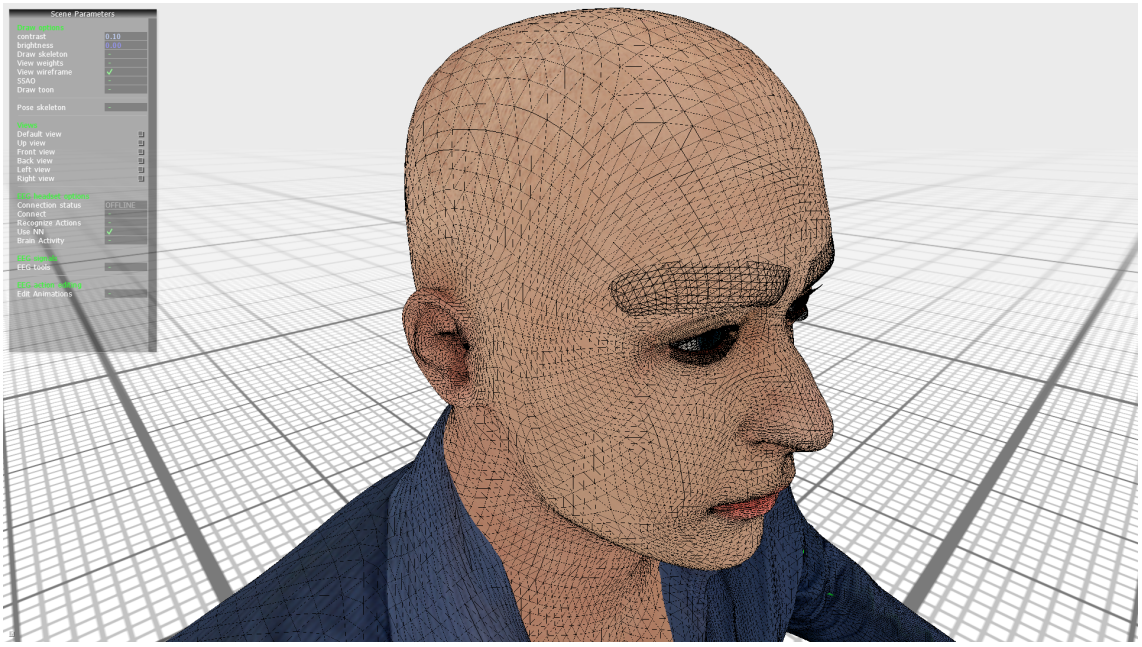
- **DrawSkeleton()**: Slouží k vykreslení kostry modelu, pro snadnější výběr kloubů při vytváření animací. Vykreslení se skládá ze dvou kroků. Jedním je vykreslení kloubů pomocí bodů a druhý je vykreslení úseček mezi nimi. Musejí tak být vytvořeny navíc dva buffery obsahující pozice kloubu a pozice kloubů ve dvojicích reprezentující úsečky. Vykreslování kostry lze zapnout při jakémkoliv jiném nastavení a daná kostra se vykresluje vždy nezávisle na hloubce uložené v hloubkovém bufferu, a tudíž ji lze vidět vždy přes geometrii lidské postavy.

- **DrawToon()**: Je určena k vykreslení nerealistického zobrazení scény tzv. cell či toon. Tato technika spočívá ve vykreslení siluet geometrie a skokovém zabarvení podle pohledově závislého phongova osvětlovacího modelu. Silueta je zjištěna cosinem uhlu mezi normálou a směrem pohledu, kde výsledná hodnota menší než 0.2 je obarvena černě a na zbytek je aplikován skokový přechod pro definovaný počet přechodů. Ukázka vykreslení geometrie tímto shaderem je na obrázku [25].

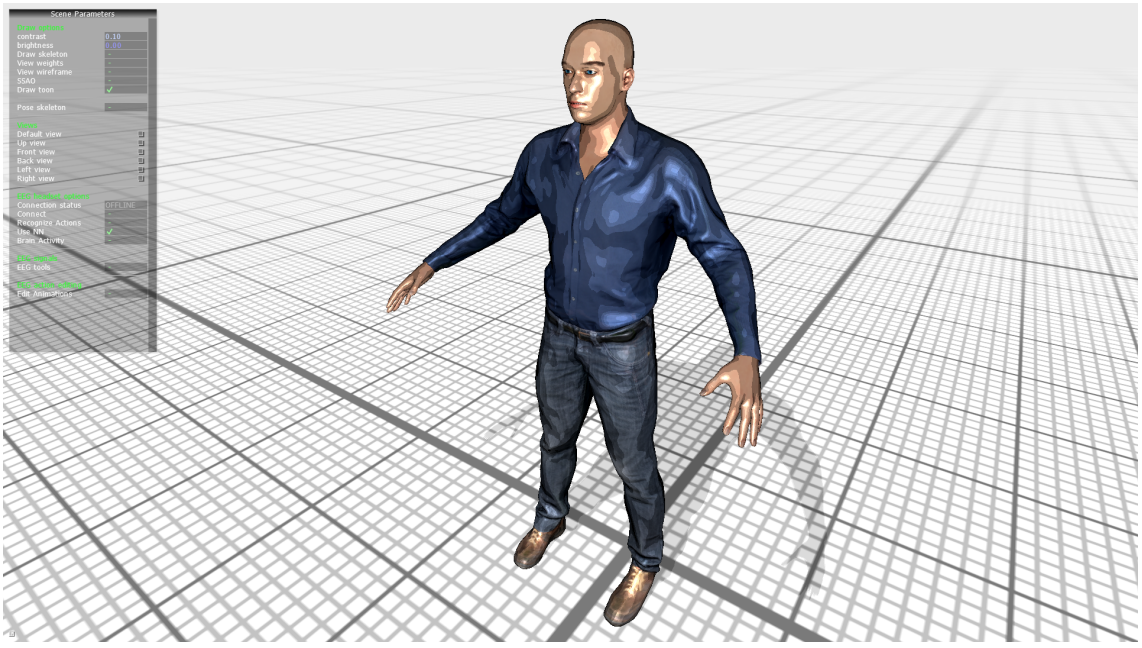
- **DrawWeights()**: Je poslední možností vizualizace ve třídě *Entity_Animated*. Pomocí této metody, a v ní použitém shaderu, lze zobrazit váhy vrcholu zvoleného kloubu. Po zvolení kloubu, kliknutím blízko jeho pozice, se obarví vrcholy ovlivněné právě tímto kloubem. Barva závisí na hodnotě váhy, která slouží jako souřadnice na 1D textuře s barevným gradientem světelného spektra viz obrázek [22].

5.3.5 Vizualizace mozkové aktivity

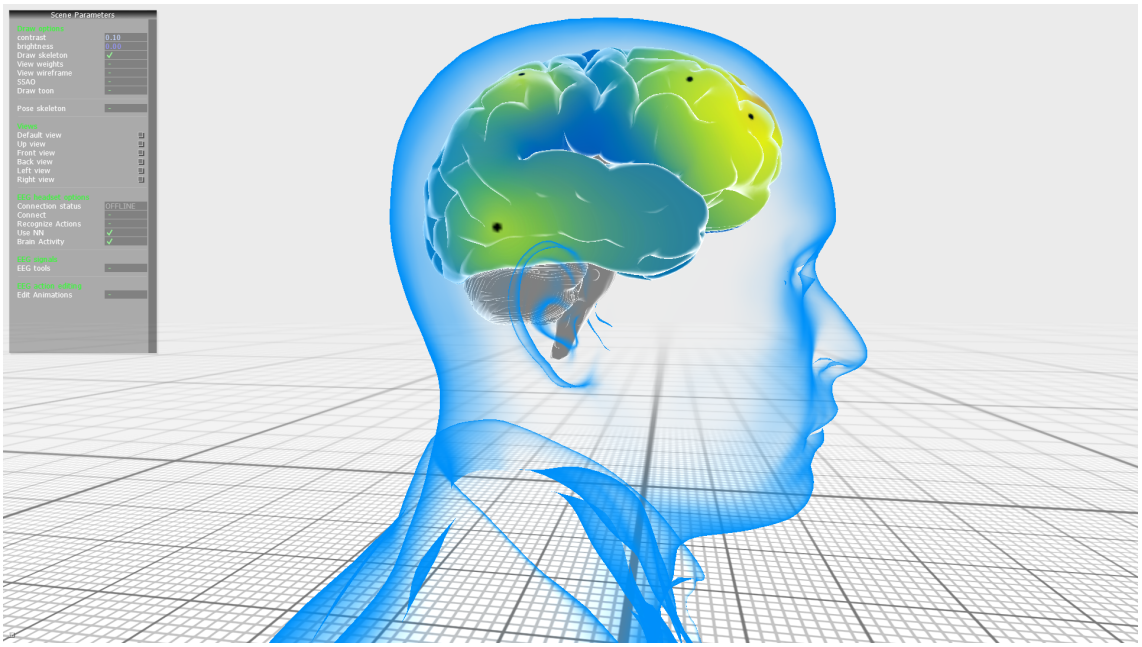
Součástí této práce bylo také vizualizovat aktivitu v jednotlivých částech mozku, které snímají senzory headsetu. K tomuto účelu posloužil věrohodný model lidského mozku, na kterém by byla tato aktivita ukázána. Tento model je k dispozici zdarma na stránce *turbosquid.com*. Pro vizualizaci bylo ovšem potřeba zjistit pozice jednotlivých senzorů na povrchu modelu. K tomuto účelu byl použit modelovací nástroj Blender, ve kterém byla vygenerována textura, na které jsou pozice reprezentovány pomocí černých kruhů. Následně byly pozice středů kruhu přepočítány na souřadnice v rozmezí 0 až 1 v obou souřadnicových osách. Pole těchto souřadnic je poté posíláno do shaderu společně s hodnotami síly signálu v jednotlivých senzorech na těchto pozicích.



Obrázek 24: Drátěný model animované postavy



Obrázek 25: Toon shader, reprezentující nerealistické vykreslování



Obrázek 26: Vizualizace mozkové aktivity

Vykreslení modelu obstarává třída *Entity_Brain*, která rovněž obsahuje metodu s názvem *setSensorPower()* pro nastavení síly mozkové aktivity v jednotlivých senzorech. Síla signálu v jednotlivých oblastech je reprezentována barvou od tmavě modré přes zelenou až po červenou. Barvy jsou reprezentovány 1D texturou obsahující právě gradient barev. Výpis [4] zobrazuje kód fragment shaderu, vypočítávající vliv jednotlivých senzorů na barvu fragmentů trojúhelníků geometrie lidského mozku. V ukázce lze vidět smyčku procházející síly všech senzorů. Pro síly větší než nula vypočítává vzdálenosti senzoru od nynější pozice na textuře a rádius. Pokud je vypočtená vzdálenost menší než rádius dosahu senzoru je vypočítáván barevný příspěvek tohoto senzoru do výsledné barvy fragmentu. Tato barva se poté smíchá s barvou nastavenou v předchozích krocích cyklu. Po vypočtení výsledné barvy se aplikuje efekt zvýraznění přechodů geometrie. To je dosaženo pomocí zjištění kosinu úhlu mezi směrem pohledu z kamery a normály fragmentu, který určuje odvrácení příslušného fragmentu od kamery. Tento koeficient byl následně škálován v rozmezí 0.4 až 1.0 a použit pro smíchání výsledné barvy pixelu s bílou barvou reprezentující výrazný přechod geometrie. Kolem mozku je rovněž vykreslena zjednodušená geometrie hlavního modelu lidské postavy pomocí stejného postupu zvýraznění přechodů geometrie pouze s využitím průhlednosti tak, aby byl mozek, umístěný uvnitř hlavy postavy, vidět. Vizualizaci mozkové aktivity lze vidět na obrázku [26].

```
for(int i = 0; i < noSensors; i++)
{
    power = sensorPowers[i];

    if (power > 0.0)
    {
        position = sensorPositions[i];
        x = position.x - texCoords.x;
        y = position.y - texCoords.y;
        radius = power * radiusAttentuation;
        coef = power * powerAttentuation;
        m = x*x + y*y - radius*radius;

        if (m < 0)
        {
            float cco = min(abs(distance(position, texCoords.xy)/radius),1.0);
            vec3 t = texture(gradientTexture, (1.0-cco)*coef).rgb;
            finalColor = mix(vec4(t, 1.0), finalColor, cco);
        }
    }
}

Fragment_color = finalColor;

float rimCon = 1.0 - max(dot(E, N), 0.0);
rimCon = smoothstep(0.4, 1.0, rimCon);


Fragment_color = mix(Fragment_color,whiteColor,rimCon)
```

Výpis 3: Kód fragment shaderu vykreslující mozkovou aktivitu v oblastech senzorů EEG headsetu.

5.3.6 SSAO

Pro vylepšení výsledného vzhledu byl implementován post-procesový efekt *SSAO*. Pro tento účel je v aplikaci pro demo scénu vytvořena třída *PostProcess*. Ta se stará o úpravu obrazu po vykreslení geometrie a pracuje tak s finálními fragmenty obrazu. K získání textury obsahující tento obraz je zapotřebí definovat framebuffer, ke kterému je přiřazena jedna či více textur. Framebuffer lze použit pomocí funkce *bindMRTfbo()* v této třídě. Tuto metodu je potřeba umístit před vykreslení geometrie tak, aby se nastavil vytvořený framebuffer pro zápis místo výchozího, systémově definovaného. Po vykreslení je potřeba opět přepnout framebuffer zpět na výchozí. Třída je tvořena tak aby mohla provést libovolný počet na sebe navazujících operací s obrazem a to díky systému dvou framebufferů, ke kterým je přiřazena barevná textura. Tyto framebuffery se po každém kroku střídají tak, že v minulém kroku vygenerovaná textura je použita jako vstup následujícího kroku. S touto texturou jsou do daného shaderu posílány rovněž textury generované během vykreslování scény do obrazu. Jsou to textury obsahující barvu, pozici fragmentů

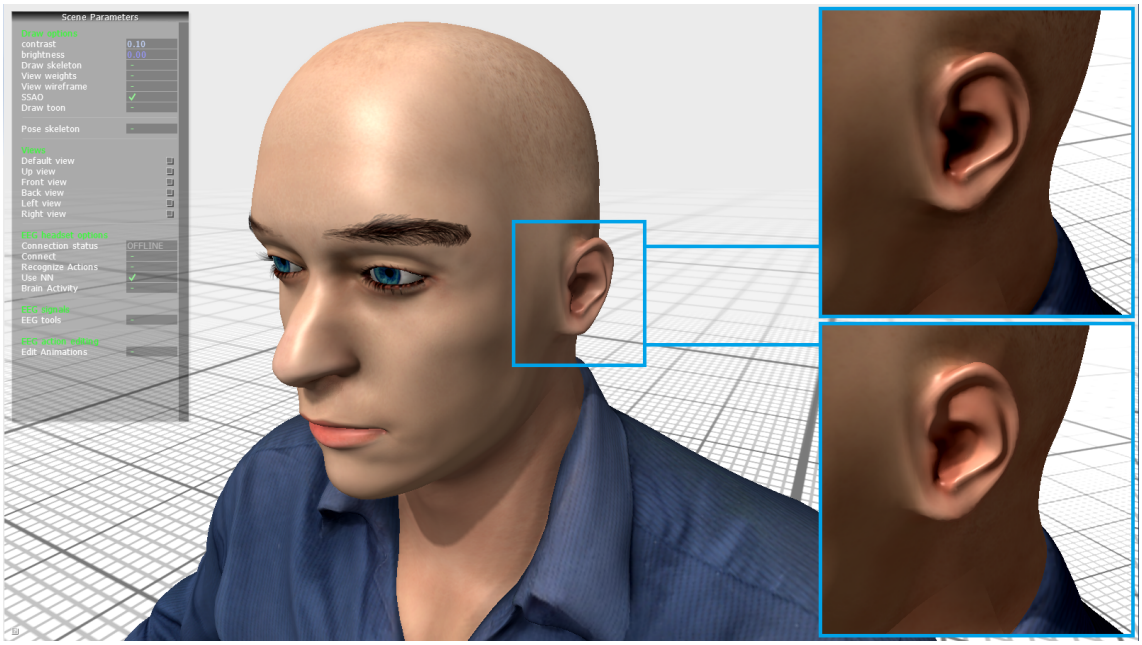
v prostoru kamery a normály fragmentů. Pomocí všech těchto textur lze vytvořit širokou škálu post-procesových efektů. Pro definování sady shaderů, které se provedou za sebou v daném pořadí, slouží jednoduchá struktura *PP_Shaders* obsahující jméno a kolekci shaderů. Lze vytvořit několik těchto post-procesových efektů složených z libovolného počtu shaderů a přepínat mezi nimi. Poslední shader v dané kolekci je použit pro vykreslení na obrazovku, a tedy do systémového framebufferu.

Tento sytém post-procesů byl použit v této aplikaci pro nastavení kontrastu a jasu výsledného obrazu a *SSAO*. Nastavení jasu a kontrastu provádí jednoduchý jednoprůchodový shader. Ten odečte od původní barvy hodnotu 0.5 a vynásobí ji dvojnásobkem nastavené hodnoty kontrastu, která je v rozmezí 0 až 1. Poté k této hodnotě přičte původní hodnotu barvy a nastavenou hodnotu jasu rovněž ve stejném rozsahu.

Druhým použitým post-procesovým efektem je *SSAO*, neboli *Screen space ambient occlusion*. Jedná se o globální metodu osvětlení scény nejčastěji doplňující výpočet lokálního osvětlení např. pomocí Phongova osvětlovacího modelu. Tento efekt slouží k aproximaci vlivu okolního prostředí na zastínění zkoumaného fragmentu a je aproximací efektu *Ambient occlusion* pro použití při vykreslování scény v reálném čase. Tento post-procesový efekt se skládá ze čtyř průchodu. První je vytvoření okluzní mapy. Druhý a třetí průchod má na starosti rozmazání této textury za použití Gaussianova filtru v horizontálním a vertikálním směru. Poslední krok je pak pouze vynásobením původní barvy okluzní mapou. Výpočet okluzní mapy probíhá následovně. Pro výpočet se používají dvě jádra pro výběr fragmentů ovlivňující okluzi každého fragmentu. Offset pro výběr prvního jádra se vypočítá následovně. Jsou určeny čtyři směry reprezentující nejbližší okolí fragmentu, pomocí hodnoty pomocné textury obsahující náhodné čísla jsou následně vypočítány jejich odrazy a vynásobeny velikostí pixelu na textuře. Druhé jádro výběru je vytvořeno otočením prvního jádra o 45 stupňů. Následně jsou tyto offsety vynásobeny poloměrem jádra a přičteny k původním souřadnicím. Takto je to provedeno čtyřikrát pro každý ze směrů s tím, že je zmenšen poloměr obou jader. Tímto způsobem je vybráno 16 prvků s různým, pseudonáhodným odstupem. Od těchto fragmentu je poté vypočítán přínos energie jakožto kosinus úhlu mezi normálou původního fragmentu a směru k vybranému fragmentu z okolí. Tato intenzita je poté vynásobena útlumem vypočítaným na základě vzdálenosti obou fragmentů. Výsledné hodnoty všech prvků jádra výběru jsou zprůměrovány a uloženy do výstupní textury. Kód implementace *SSAO* je uveden níže. Na obrázku[27] je vyobrazen vliv SSAO na výsledný vzhled scény.

```
float SamplePixels (vec3 srcPosition, vec3 srcNormal, vec2 uv)
{

    vec3 dstPosition = texture(positionTexture, uv).xyz;

    vec3 positionVec = dstPosition - srcPosition;

    float intensity = max(dot(normalize(positionVec), srcNormal) - OccluderBias, 0.0);

    float lng = length(positionVec);
    float attenuation = 1.0 / (Attenuation.x + (Attenuation.y * lng));

    return intensity * attenuation;
}

void main()
{
    vec3 srcPosition = texture(positionTexture, tCoords).xyz ;
    vec3 srcNormal = texture(normalTexture, tCoords).xyz;
    vec2 randVec = normalize(texture(randomTexture, tCoords).xy*2+1);

    float occlusion = 0.0;

    vec2 kernel[4];
    kernel[0] = vec2(0.0, 1.0);
    kernel[1] = vec2(1.0, 0.0);
    kernel[2] = vec2(0.0, -1.0);
    kernel[3] = vec2(-1.0, 0.0);


    for (int i = 0; i < 4; i++)
    {
        vec2 k1 = reflect(kernel[i], randVec);
        vec2 k2 = vec2(k1.x * Sin45 - k1.y * Sin45,k1.x * Sin45 + k1.y * Sin45);

        k1 *= TexelSize;
        k2 *= TexelSize;

        occlusion += SamplePixels(srcPosition, srcNormal, tCoords + k1 * kernelRadius);
        occlusion += SamplePixels(srcPosition, srcNormal, tCoords + k2 * kernelRadius * 0.75);
        occlusion += SamplePixels(srcPosition, srcNormal, tCoords + k1 * kernelRadius * 0.5);
        occlusion += SamplePixels(srcPosition, srcNormal, tCoords + k2 * kernelRadius * 0.25);
    }

    occlusion /= 16.0;

    FragColor = vec4(occlusion,occlusion,occlusion,1.0);


}
```

Výpis 4: Kód fragment shaderu vypočítávající okluzní mapu efektu SSAO



Obrázek 27: SSAO s detailem, nahoře zapnuto a dole vypnuto

5.3.7 Testování aplikace

Vývoj a testování této aplikace probíhal na počítačové sestavě A uvedené v tabulce [2]. Pro vývoj bylo použito vývojové prostředí Microsoft Visual Studio 2013. Grafická scéna byla implementována v OpenGL v souladu se specifikací verze 4.5. V knihovně pro správu headsetu bylo použito Qt ve verzi 5.4. Pro účely testování byl použit i počítač na kterém byla vytvořena knihovna zajišťující přístup k headsetu a rozpoznávání akcí, a to notebook Lenovo ThinkPad T520 jehož parametry jsou uvedeny v tabulce [3] pod názvem sestava B.

Procesor	Intel(R) Core(TM) i5-4430 3GHz
Operační paměť	8 GB DDR3
Grafický procesor	NVIDIA GeForce GTX760 2GB GDDR5
Operační systém	Microsoft Winfows 7 Professional 64-bit

Tabulka 2: Parametry sestavy A

V tabulkách [4] a [5] lze vidět výsledky testů na obou testovaných sestavách s různým nastavením vizualizace. Ve všech případech byl měřen počet snímků za sekundu, který je pro plynulost vizualizace klíčový. Jako základní vykreslení je zvoleno vykreslení lidské postavy bez stínů, vlivu animací či vykreslení její kostry. Pro měření vlivu post-procesových efektů bylo požito základní nastavení se zapnutými stíny a vlivy animací. Do měření post-procesových shaderů byla zahrnuta změna výpočetního jádra pro každý fragment. Jednotlivé velikosti jsou označeny číslem za názvem SSAO. Pro všechny

Procesor	Intel(R) Core(TM) i5-2520M 2.5GHz
Operační paměť	8 GB DDR3
Grafický procesor	NVIDIA NVS4200M 1GB GDDR3
Operační systém	Microsoft Winfows 7 Professional 64-bit

Tabulka 3: Parametry sestavy B

případy měření byla vypnuta i komunikace mezi knihovnou pro získávání dat headsetu a aplikací pro správu headsetu tak, aby bylo eliminováno zpomalení. Při zapnutí aplikace pro správu headsetu bylo zaznamenáno pouze velmi nízký pokles okolo 5 snímků za sekundu. To ovšem neplatilo při přepnutí na záložku *Senzors*, kde se ukázala mezera optimalizace Qt, kdy došlo ke snížení počtu snímků za sekundu přibližně o 50, při zapnutí základní vizualizace se stíny a animacemi.

Nastavení	**Sestava A [FPS]**	**Sestava B[FPS]**
Základní	1192	136
Základní + animace	900	203
Základní + stíny	451	129
Základní + kostra	1178	60
Základní + animace + stíny	392	51
Drátěný model	312	45
Váhy	510	69
Toon	435	63

Tabulka 4: Výsledky měření výkonu jednotlivých vizualizací

Shader	**Sestava A [FPS]**	**Sestava B[FPS]**
Jas a kontrast	382	41
Jas a kontrast + SSAO 16	138	12
SSAO 16	149	15
SSAO 32	115	10
SSAO 64	78	7

Tabulka 5: Výsledky měření výkonu post-procesových shaderů

Z naměřených výsledků lze vidět jak rozdíl mezi testovanými sestavami, tak i výkonové rozdíly jednotlivých operací. Lze vidět, že i když byly použity matice k deformaci animované postavy, je propad počtu snímku za sekundu téměř čtvrtinový. Ještě větší zátěž představují stíny s použitím PCF pro zjemnění. Zde je propad víc než poloviční a stíny tak představují nejnáročnější operaci ve scéně. To je částečně způsobeno použitím velmi přesné textury pro generování hloubkové mapy a tím, že podrobnou geometrii lidské postavy bylo nutné vykreslit pro jeden snímek dvakrát. Naopak velmi malé rozdíly jsou v případě vykreslení kostry. Použití Toon shaderu je výkonově náročné velmi

podobně jako vykreslení základní metodou, a to je dáno podobnými výpočetními operacemi v shaderech. Vykreslení drátěného modelu je přibližně o čtvrtinu náročnější než vykreslení základního modelu se stíny. To je dáno opět vykreslením geometrie dvakrát pro jeden snímek. Rozdíl však není tak velký, jak by se mohlo zdát. To je zapříčiněno méně náročnými shadery oproti základnímu vykreslování.

U post-procesového shaderu lze vidět mírný pokles počet snímků za sekundu, a to okolo 10. U takhle jednoduchého shaderu lze připsat větší část tohoto poklesu režii přepínání framebufferů a generování přesných vstupních textur. Při použití SSAO lze vidět znatelný propad počtu snímků za sekundu daný vysokou náročností shaderu pro výpočet okluzní mapy a také fakt, že je použito 4 průchodové posloupnosti shaderů zahrnující opět nutnou režii framebufferů. Horizontální a vertikální rozmazaní obrazu pomocí Gaussianova filtru se však ukázalo velmi úsporným řešením. Při odstranění použití těchto shaderů bylo zaznamenáno jen minoritní zvětšení snímků za sekundu a to v rámci jednotek. Při násobení velikosti jádra výpočtu okluzní mapy byl zaznamenán pokles vždy o přibližně 35 snímků za sekundu.

6 Závěr

V teoretické části této práce byly popsány principy EEG a EEG signálu samotného a druhů vln, které se v něm nacházejí. Poté byl popsán mozek jako takový včetně nejdůležitější části pro tuto práci a to koncový mozek a jeho laloky. Dále jsou popsány BCI systémy s důležitými historickými milníky ve vědeckých experimentech jak u zvířat, tak i lidí pomocí různých zařízení. V teoretické části byly popsány také systémy mapování pozic senzorů na lebku, a to především systém 10/20, který byl použit u headsetu Emotiv EPOC, který byl následně popsán.

Cílem práce bylo vytvořit aplikaci demo scény pro vizualizaci motorických stavů lidského těla rozpoznaných headsetem snímající EEG signály z různých oblastí mozku. Rovněž byla implementována aplikace pro správu EEG headsetu. Pro snímání EEG vln byl použit headset Emotiv EPOC, avšak byl kladen důraz na univerzálnost obou aplikací tak, aby bylo možno použít jakékoliv zařízení. K obsluze headsetu, získávání dat, jejich zpracování a rozpoznání akcí byla použita knihovna. Byl kladen důraz také na to, aby tyto aplikace byly od sebe, a taktéž od knihovny pro získávání dat z headsetu, oddělené a pracovali jako samostatné, nezávislé části. K implementaci těchto zařízení posloužil jazyk C++, technologie Qt a OpenGL. Pro reprezentaci lidské postavy byla implementována 3D vizualizace pomocí knihovny OpenGL. K vygenerování geometrie byl použit program MakeHuman. Jako přenosový formát byl použit formát MD5. Deformace geometrie probíhala pomocí matic na GPU. V aplikaci demo scény byly rovněž implementovány různé styly vizualizace pokrývající potřeby práce včetně vizualizace mozkové aktivity v jednotlivých oblastech snímaných senzory. Byl implementován i editor animací rozpoznávaných akcí, včetně polohování modelu a vytváření klíčových snímku, reprezentující přechodné stavy animace. V aplikaci pro správu headsetu bylo implementováno vše potřebné k správě headsetu, včetně vizualizace kvality příjmu signálu jeho senzorů, stavu baterie či signálu připojení. Byla zde také implementována funkcionalita spojená s nahráváním záznamů, vybíráním vzorků v něm a ukládáním akcí. Byl vytvořen kompletní a intuitivní systém označování vzorků přímo v grafech zobrazující záznam EEG signálu. Obě aplikace tak tvoří funkční celek obsahující vše potřebné k vytváření a vizualizaci akcí reprezentující motorické stavy lidského těla. Závěr praktické části je věnován testování jednotlivých nastavení vizualizace na dvou počítačových sestavách.

Vypracování práce obsahuje všechny body obsažené v zadání práce. Aplikaci by bylo možné dále rozšířit o další prvky vizualizace, případně další práci s animovanou postavou. Bylo by tak možné rozšířit vizualizaci demo scény o pokročilé techniky nasvícení kůže či látek oblečení nebo např. implementace dynamických vlasů.

Jakub Rodzenák

7 Reference

[1] Saeid Sanei, J. A. *Chambers: EEG Signal Processing, Wiley-Interscience*, 1 edition (September 11, 2007), ISBN-13: 978-0470025819

[2] Berger, Hans, *Hans Berger on the electroencephalogram of man : the fourteen original reports on the human electroencephalogram,* Elsevier Pub. Co., 1969. ISBN-13: 978-0444407399 ISBN-10: 0444407391

[3] Haas, L. F., *Hans Berger (1873–1941), Richard Caton (1842–1926), and electroencephalography,* 2003. PMID: 12486257

[4] Niedermeyer, E., Silva, F.L., *Electroencephalography: Basic Principles, Clinical Applications, and Related Fields,* 2004. ISBN 0-7817-5126-8

[5] Georgopoulos, A., Lurito, J.; Petrides, M.; Schwartz, A; Massey, J., *Mental rotation of the neuronal population vector,* 1989. PMID: 2911737

[6] Freeman, W., Rodrigo Quian Quiroga, *Imaging Brain Function With EEG: Advanced Temporal and Spatial Analysis of Electroencephalographic Signals* 2012. ISBN-13: 978-1461449836 ISBN-10: 1461449839

[7] Libenson, Mark H., *Practical Approach to Electroencephalography*, c2009. ISBN-13: 978-0750674782 ISBN-10: 0750674784

[8] Edward Angel: *Interactive Computer Graphics*, ISBN-10:032153586 (2009)

[9] *Seminární práce z biofyziky*[online]. 2004.
<http://ftplf2.agarek.com/fyzio/prvak/biofyzika/semin/helcas_eeg>.

[10] *Brain-computer interface*[online]. 2014. <http://en.wikipedia.org/wiki/Brain

[11] RNDr. Vlasta Lungová Ph.D. *Stavba a funkce lidského mozku*[online]. Katedra zoologie a ornitologie, PřF UP Olomouc.[cit. 23.11.2012]. <http://pfyziollfup.upol.cz/castwiki/?p=3265>.

[12] *Khronos group*[online]Khronos Group. [cit. 14.4.2015]. <https://www.khronos.org/>.

[13] *OpenGL*[online]Khronos Group. [cit. 14.4.2015]. <https://www.opengl.org/>.

[14] Wright, Richard S., Graham Sellers, a Nicholas Haemel. *OpenGL superbible : comprehensive tutorial and reference*. Upper Saddle River, NJ: Addison-Wesley, 2014. ISBN-13: 978-0321902948 ISBN-10: 0321902947

[15] Shreiner, Dave. *OpenGL programming guide : the official guide to learning OpenGL, version 2*. Upper Saddle River, NJ: Addison-Wesley, 2006. ISBN-13: 978-0321773036 ISBN-10: 0321773039

[16] Cozzi, Patrick, and Christophe Riccio. *OpenGL insights*. Boca Raton, FL: CRC Press, 2012. ISBN-13: 978-1439893760 ISBN-10: 1439893764

[17] *Qt*[online]The Qt Company. [cit. 13.4.2015]. <http://www.qt.io/>.

[18] *Qt*[online]Wikipedia. [cit. 10.5.2015]. < http://en.wikipedia.org/wiki/Qt_

[19] *Qwt*[online]Uwe Rathmann. [cit. 10.5.2015]. < http://qwt.sourceforge.net/>.

[20] *Makehuman*[online]MakeHuman team . [cit. 15.4.2015]. <http://www.makehuman.org/>.

[21] *Blender*[online]Blender . [cit. 15.4.2015]. <http://www.blender.org/>.

[22] *Emotiv*[online]Emotiv . [cit. 15.4.2015]. <https://emotiv.com/>.

[23] *Emotiv Epoc*[online]Emotiv . [cit. 14.4.2015]. <https://emotiv.com/epoc.php>.

[24] *EASYCAP*[online]Easycap . [cit. 15.4.2015]. <http://www.easycap.de/easycap/index.htm>.

[25] *Electroencephalography*[online]. [cit. 15.4.2015]. <http://www.bem.fi/book/13/13.htm>.

[26] *Ten Unusual Experiments in the Name of Science – Getting Cat Vision*[online]. [cit. 1.6.2012]. <http://www.united-academics.org/magazine/badscience/ten-unusual-experiments-in-the-name-of-science-getting-cat-vision/>.

[27] *ActiCAP*[online]Bionic. [cit. 6.2.2011]. <http://www.bionic.es/product/acticap/>.

[28] *Helicopter takes to the skies with the power of thought*[online]Phys. [cit. 4.6.2013]. <http://phys.org/news/2013-06-helicopter-power-thought-video.html/>.

[29] *Pitt Team Publishes New Findings from Mind-Controlled Robot Arm Project*[online]UPMC/University of Pittsburgh Schools of the Health Sciences. [cit. 4.6.2013]. <http://www.upmc.com/media/media-kit/bci/pages/default.aspx>.

A Příloha na CD

Součástí diplomové práce je CD obsahující elektronickou formu práce, zdrojové soubory v podobě projektu pro Microsoft Visual Studio 2013 a zkompilovanou aplikaci, připravenou pro spuštění.

Obsah přiloženého CD:

/Aplikace
/Diplomova_prace_Jakub_Rodzenak.pdf
/Projekt_MS_VS_2013
/Readme.txt